Bildmustererkennung mit Hilfe spärlich codierter Assoziativmatrizen (SpaCAM) am Beispiel von Kfz-Kennzeichen

Vom Fachbereich III – Informations- und Kommunikationswissenschaften - der Universität Hildesheim

zur Erlangung des Grades eines Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigte

D i s s e r t a t i o n

von

Detlef Romberger

aus Kassel

Berichterstatter: Prof. Dr. H.-J. Bentz
Prof. Dr. M. Overhoff

eingereicht am: 8. Juni 2004
mündliche Prüfung am: 14. Dezember 2005

Danksagung

Zunächst einmal bedanke ich mich als externer Doktorand bei Herrn Prof. Dr. Bentz für die Überlassung des Themas und die stets angenehme und fruchtbare Zusammenarbeit; ebenfalls danke ich Herrn Prof. Dr. Schwarzer für die Betreuung.

Herrn Prof. Dr. Overhoff von der Fachhochschule in Gelsenkirchen danke ich für die Unterstützung, für wertvolle Hinweise und für die Übernahme des externen Gutachtens.

Der Firma Bitzer GmbH in Hildesheim (www.bitzer-waage.de) sage ich Danke für die Überlassung einiger besonderer Bildvorlagen und für Anregungen aus der Praxis.

Dank gilt auch meiner Familie für die Geduld und die Unterstützung.

Erklärung

Hiermit versichere ich an Eides statt, dass ich die Abhandlung mit **Schwerpunkt** in der **Informatik** selbständig und ohne unerlaubte Hilfsmittel verfasst und die benutzten Hilfsmittel vollständig angegeben habe.

Korbach/Hildesheim, den 8. Juni 2004

Kurzfassung

In dieser Arbeit geht es um Mustererkennung für „standardisierte“ Schriftbilder. Sie baut auf früheren Untersuchungen der Projektgruppe „Neuronale Netze – SpaCAM“ an der Universität Hildesheim auf, wobei es damals um die Wiedererkennung handgeschriebener Unterschriften auf Überweisungsvordrucken ging, also um ein Erkennungsproblem, bei der es in den Daten typischerweise kein Standard-Referenzmuster gibt. Da kein Mensch (physikalisch) identische Unterschriften produzieren kann, muss die Referenz(menge) aus einer Sammlung gegebener Samples „willkürlich“ festgesetzt werden. Grundlage für jene Problembehandlung war die Verwendung von „Spärlich Codierten Assoziativmatrizen SpaCAM“, die man als eine spezielle Variante der künstlichen neuronalen Netze auffassen kann. Die SpaCAM-Technik hat sich dabei – wie auch schon beim Einsatz in anderen Problembereichen - als besonders vorteilhaft in Bezug auf Fehlertoleranz, Robustheit und Geschwindigkeit erwiesen.

Das Ziel der vorliegenden Arbeit ist es zu untersuchen, ob sich diese Technik auch zum Erkennen von Bildmustern eignet, bei denen die Referenzmuster bereits (extern) vorgegeben sind. Als Anwendungsfall wurden „Autokennzeichen“ gewählt, auch weil es hierfür eine konkrete Problemstellung aus der Praxis gab, mit Zusatzparametern, die den Komplexitätsgrad des Erkennungsprozesses bereicherten. In Bezug auf die SpaCAM-Technologie soll geprüft werden, welche Merkmale bzw. Merkmalsgruppen für die Codierung von Zeichen besonders gut geeignet sind, insb. beim Lernen aus Schriftarten, um die Erkennung praktikabel zu machen.

Für die Analyse und Diskussion der Phänomene wird eine Arbeitsumgebung entwickelt, die auf die sich ergebenden unterschiedlichen Teilaufgaben ausgerichtet ist, also Probleme der Vorverarbeitung (u.a. Grauwerte, Monochrome, Segmentierung), der Speicherprozesse (u.a. Merkmalscodierung, Referenzmengen) aber auch der Nachbestimmung (z.B. durch fehlertoleranten Datenbankabgleich) behandelt und zudem einen Spielraum für weitere Anpassungen bietet. Zur Verifizierung der Tauglichkeit des Ansatzes und zur Überprüfung der Prototyp-Implementation werden alle Teilmodule auf das konkrete Problem der Kfz-Kennzeichen-Erkennung angewandt.

Die Software ist in Pascal („Delphi 5 Professional“) geschrieben und enthält neben den eigentlichen Mustererkennungs- auch umfangreiche Analyse- und Visualisierungsfunktionen. Daher eignet sie sich auch für Schulungszwecke und Präsentationen.

Im konkret zugrunde liegenden Anwendungsfall entstammt die Menge der relevanten Kennzeichen einer Datenbasis, die neben dem authentischen Kennzeichen auch Sekundärinformationen (Fahrzeugdaten, Halter etc) enthält, welche für nachfolgende Entscheidungsprozesse von Bedeutung sind. Hier wurde im Hinblick auf den Einsatz in der Praxis zusätzlich eine ebenfalls auf der SpaCAM-Technik basierende Schnittstelle zwecks fehlertoleranter Suche nach Sekundärinformationen in der „gewöhnlichen“ Datenbank implementiert. Insofern könnten die im Nachgang ermittelten Informationen unter Umständen beim vorherigen Erkennungsprozess mithelfen, Fehler oder Unentscheidbarkeiten zu verringern. Für die Technologie bedeutet das, dass in den separaten Phasen unterschiedliche Merkmalsmengen und Codevektoren gefunden werden müssen, deren Zusammenspiel aber eine Verbesserung der Endergebnisse bewirken kann. Auch diese Komposition soll beispielhaft dargestellt werden.

Mit der Fragestellung der vorliegenden Arbeit befindet man sich nicht weit entfernt von aktuell diskutierten Sicherheits- oder Automatisierungsvorhaben. Man denke zum Beispiel an die Verwendung biometrischer Merkmale (Fingerabdrücke, Iris im Auge, Gesichtsform usw.) als Zugangskontrolle an Flughäfen, Schließfächern, Laboratorien usw. oder an die automatische Mauterhebung von Fahrzeugen u.v.m. Insofern besteht nicht nur ein akademisches Interesse an den Einsatzmöglichkeiten neuer Technologien, sondern auch ein organisatorisches bzw. gesellschaftliches, welches als Herausforderung an die Wissenschaftler empfunden werden kann.

Abstract

This thesis deals with the recognition of standardised type faces and is based on the research carried out by the “Neural networks – SpaCAM” group at the University of Hildesheim. Their research was into recognizing signatures on bank transfer forms. In this case, the learned references (that is, signatures) were defined by special sample instead of an external norm - because nobody is able to write identical, standard signatures. The “Sparsely Coded Associated Memory (SpaCAM)”, a special variation of an artificial neural network, was used to solve this problem. In this and other applications the SpaCAM has proved itself to be fault tolerant, robust and fast.

The aim of this paper is to determine whether this technique is appropriate to recognise patterns where an externally norm defines the references. The Vehicle registration plate domain was chosen because there was a practical application with additional parameters which increase the complexity of the recognition process. In this domain, the reference object (a font) is predetermined. The investigation into the SpaCAM-Technique shall verify, which characteristics or set of characteristics are particularly appropriate for coding of characters to make recognition practical.

A workbench is developed to investigate the various subtasks. These are special pre-processing concepts (such as changing to grey scale, monochrome and segmentation), memory processes (such as coding of characteristics or the set of references) and post processing (e.g. the comparison in a fault tolerant database) which offers a rich set of tuning parameters. The problem chosen is intended to test the efficacy of the theory and the prototype implementation. In addition it requires the use of all workbench modules.

The software, written in Pascal (Delphi 5 Professional), includes functions for pattern recognition and a rich set of analysis and visualisation tools. Therefore, it can also be used for training and presentations.

In the chosen problem domain the set of relevant vehicle licence plates including secondary information (such as the user or the type of a car) is stored in a local database. This secondary information may be used in support of the recognition process. In addition, for practical applications, an interface, also based on SpaCAM, may be used to find secondary information on a fault-tolerant basis in the (traditional) local database. It may be that this secondary process can help to decrease the main recognition phase error rate. From the technological standpoint, this means that to improve the final result, different sets of characteristics and code vectors must be found for the separate phases. This composition shall also be illustrated by means of example.

The subject matter of this paper is not far away from current projects discussing security or task automation. Examples include recognition of biometric characteristics for access control to airports, safe deposit boxes, laboratories or toll collection for vehicles. Biometric characteristics include fingerprints, facial patterns and iris scans. There is not only academic interest in the application of these new technologies, but also an organisational and social interest and this represents a challenge to researchers.

Inhaltsverzeichnis

Abbildungsverzeichnis

Programmcodeverzeichnis

Bilderverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Zielsetzung

Als Teilgebiet automatisierter **Mustererkennung** entstand im Rahmen dieser Dissertation mit dem **Schwerpunkt in der Informatik** eine Software-Experimentierumgebung, die erstens die Voraussetzungen für den Erkennungsprozess bei Autokennzeichen analysieren lässt und zweitens eine Schnittstelle für eine mögliche Erweiterung der Software bietet, in einer Datenbank weitere Informationen zu dem identifizierten Fahrzeug zu finden (z.B. Fahrzeughalter, Fahrzeugtyp, etc.). Da die Erkennung des Autokennzeichens fehlerbehaftet sein kann, muss die Suche in der Datenbank fehlertolerant sein.

In der Vergangenheit wurden bereits einige Projekte erfolgreich mit der Idee, neuronale Netze in einer durch spärlich codierte Assoziativmatrizen[1] erweiterten Form als Technik zu nutzen, durchgeführt (s. insb. Kapitel 2.14). Dabei hat sich die SpaCAM-Technik als sehr günstig bzgl. Fehlertoleranz, Robustheit und Geschwindigkeit erwiesen. In dieser Arbeit soll nun untersucht werden, ob und wie sich diese Vorteile nutzbringend auf die Erkennung von Autokennzeichen übertragen lassen. Dabei soll auch geprüft werden, welche Merkmale, insb. beim Lernen der Referenzobjekte aus Schriftfonts, eingesetzt werden sollten, um die Erkennung zu optimieren.

Die Software ist mit umfangreichen Übungs-, Analyse- und Visualisierungstools ausgestattet, so dass sie sich durchaus auch zur Präsentation und Darstellung der SpaCAM-Technik, z.B. in Vorlesungen und Seminaren, auf Tagungen, Messen etc. eignet.

[1] Sparsely Coded Associative Memory; kurz **SpaCAM**

1.2 Aufbau der Arbeit

Im anschließenden Kapitel 2 werden die Theorien der Vorverarbeitung, Segmentierung, Merkmale, Bildmustererkennung und der fehlertoleranten Suche in Datenbanken (allg. Retrieval) soweit dargestellt, wie es für die Problemstellung dieser Arbeit relevant ist. Weiterhin werden auszugsweise andere als das hier vorgestellte Verfahren zur Autokennzeichenerkennung angesprochen. Die Implementierung der SpaCAM-Methode in eine Delphi-Applikation wird in Kapitel 3 incl. vieler Beispiele ausführlich beschrieben; ausgewählte Datenstrukturen werden im Quellcode dokumentiert. Kapitel 4 schließlich soll einen Ausblick liefern, wie die hier vorgestellten Techniken weiter entwickelt und für andere Anwendungsfälle nutzbar gemacht werden können.

2 Bildmustererkennung

2.1 Einleitung

Mustererkennung durch Computer ist ein aktuelles Thema. Man denke nur an die Erkennung biometrischer Merkmale z. B. eines Handabdrucks zum Prüfen der Zugangsberechtigung zu den Schließfächern einer Bank in den USA [**WiWo2003b**, Seite 72], der Adern auf dem Handrücken [**WiWo2004a**, Seite 86] oder von Fingerabdrücken, z.B. als Schlüsselersatz beim Zugang in Kleinbetriebe oder Haushalte [**WiWo2003a,** Seite 70] oder in der Kryptologie [**WiWo2004b**, Seite 86].
Auch das maschinelle „Zusammenpuzzeln“ von ca. 600 Millionen Papierschnipseln, Überreste von nach der Wende von den Geheimdiensten geshredderten Stasi-Geheimakten, ist eine mögliche Applikation. Weiterhin ist das Erkennen von Barcodes aus dem täglichen Leben nicht mehr wegzudenken.

Die Erkennung von Schriften ist ein altes Thema. Zunächst wurden standardisierte Formate vorgegeben (Stichwort OCRA/ICR) [**SCHÜ1977**], später kamen dann Klarschriftleser, insb. für Druckbuchstaben, hinzu [**MART1997**]. Mit dieser Technik ist es möglich, handgeschriebene Formulare per Computer maschinell einzulesen. Auch besteht damit die Möglichkeit, Dokumente, die nur in Papierform existieren, maschinell einzulesen, ggf. weiterzubearbeiten oder elektronisch zu archivieren.

Bereits 1997 wurde ein System entwickelt, mit dem Unterschriften auf Banküberweisungen erkannt werden können [**BECK1997**]. Damit wird der Bankmitarbeiter bei der Prüfung der Legitimation des Kunden durch den Rechner entlastet bzw. unterstützt. In China wurde der erste Organizer vorgestellt, der chinesische Handschriften erkennen kann [**WiWo2003c**, Seite 51].

Das Erkennen von Autokennzeichen ist aus vielerlei Gründen wünschenswert. So wird z.B. in der Tiefgarage des Flughafens von Palma de Mallorca beim Einfahren das Autokennzeichen fotografiert und beim Ausfahren des Fahrzeugs geprüft.

Nach der Novelle des hessischen Polizeigesetzes ist geplant, die Polizeibeamten mit Lesegeräten auszustatten, die es ermöglichen, Kfz-Kennzeichen zu erkennen und sie automatisch mit den Daten im Fahndungsbestand abzugleichen [**HES2003**].

Allein diese wenigen Beispiele zeigen, dass es um die beiden folgenden grundsätzlichen Begriffe geht: Die **Erkennung** von **Mustern**.
Mit anderen Worten: Wie beschreibt man Muster so, dass sie entsprechend einer bestimmten Anforderung maschinell erkannt werden können?

Da sich Muster, wie z.B. Unterschriften, Gesichter, Texte, Bilder, Fingerabdrücke usw. in ihrer Struktur außerordentlich unterscheiden, wird Erkennung dadurch möglich, dass der Rechner im Vorfeld bestimmte Merkmale der später zu erkennenden Muster „lernt“ und anhand dieser Merkmale die Muster „wieder erkennt“.
Schlüsselt man nach Teilproblemen auf, so ergibt sich:

- Wie und in welcher Form liegt das zu erkennende Muster vor? Ist es bereits digital vorhanden (z.B. als digitales Foto oder auch digitales Video) oder muss es zunächst mit Hilfe eines Scanners in eine digitale Form gebracht werden?
- In welcher (optischen) Qualität liegt das zu erkennende Muster vor, und muss ggf. eine Vorverarbeitung durchgeführt werden?
- Soll das gesamte Muster erkannt werden oder nur Teile davon? Müssen z.B. bei der Texterkennung eines Zeitungsausschnitts zunächst die Bereiche mit Text von denen mit Bildern, Werbung etc. getrennt werden?
- Sollen Muster in ihrer Gesamtheit oder durch Aufteilung in mehrere (gleichartige) Teile erkannt werden? Soll also z.B. bei Texten der Textfluss in seiner Gesamtheit (z.B. bei Unterschriften) oder einzelne Textzeichen, wie Buchstaben, Ziffern, Sonderzeichen etc, erkannt werden?
- Sind die zu erkennenden Muster farbig, grau oder monochrom?
- Welche Merkmale sind eindeutig und diskriminierend genug, das Muster möglichst sicher zu erkennen?
- Wie und in welcher Form sollen diese Merkmale verwaltet und gespeichert werden, damit das Erkennen möglichst effizient (Echtzeitverarbeitung) implementiert werden kann?

Bereits aus dieser Zusammenstellung wird deutlich, dass es „die“ Mustererkennung nicht gibt und vermutlich auch nicht geben wird. Man wird also mit einer Vielzahl von Lösungsansätzen arbeiten, wobei man immer wieder (überrascht) feststellt, dass die Lösung eines speziellen Problems auf andere Fälle übertragbar ist.

In dieser Arbeit geht es um die Erkennung von Autokennzeichen mit Hilfe spezieller neuronaler Netze.

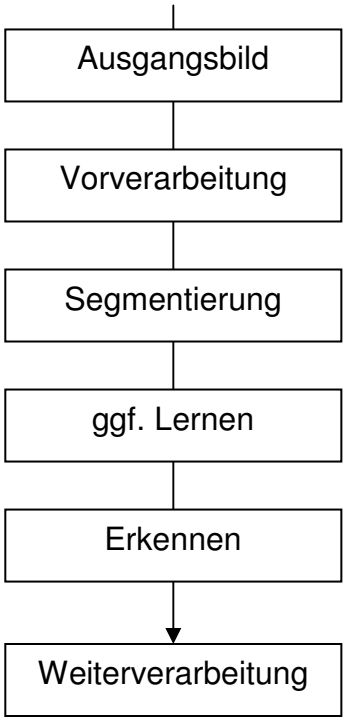
2.2 Problembeschreibung

Als ***Ausgangsbild*** liegt ein digitales Foto eines Fahrzeugs (PKW, LKW, Motorrad etc.) vor. Dieses Foto kann online von einer Kamera oder Webcam permanent abgegriffen werden oder es liegt als Datei in einem der üblichen Formate (z.B. Bitmap (*.bmp); JPEG (*.JPG) etc.) auf einem externen Speicher. Das Ausgangsbild wird in den seltensten Fällen nur das Kfz-Kennzeichen enthalten, sondern es wird eine mehr oder weniger große, i.d.R. farbige Aufnahme der Front- oder Heckpartie des Fahrzeuges sein. Aus diesem Bild muss das Kennzeichen separiert werden.
Die optische Qualität dieser Aufnahme wird sehr stark schwanken durch Einflüsse wie Sonne, Regen, Schnee, Nebel, Schmutz, Tages-, Nachtaufnahme, Unschärfe, Drehungen, Nah-, Fernaufnahme, Weitwinkel, Zoom usw. Deshalb muss zunächst eine **Vorverarbeitung** stattfinden wie z.B. Aufhellung, Ausrichtung in die Waagerechte etc. Trotzdem dürfte es kaum möglich sein, das Kennzeichen, wenn man es in seiner Gesamtheit erkennen wollte, direkt mit gespeicherter Information zu vergleichen. Man denke nur an ein stark verschmutztes Kennzeichen. Dazu kommt noch, dass Schraubenköpfe und weitere Markierungen ein Erkennen schwierig, wenn nicht gar unmöglich machen.
Man wird also versuchen, jedes einzelne Zeichen (Buchstabe/Ziffer) für sich zu erkennen, um daraus das Autokennzeichen als String zusammenzusetzen. Somit muss das Ausgangsbild in die Zeichen des Kfz-Kennzeichens „zerlegt“ werden, mit dem Risiko, dass dabei natürlich auch andere Buchstaben/Ziffern gefunden werden, die nicht zum Kfz-Kennzeichen gehören wie z.B. Beschriftungen oder Werbung auf den Fahrzeugen. Diese müssen dann später entfernt werden.
Nach dem Zerlegen, auch ***Segmentieren*** genannt, erhält man die ***Segmente*** als einzelne Bereiche des zerlegten Ausgangsbildes. Die Segmente „umrahmen“ somit

einen i.d.R. rechteckigen Teil des Ausgangsbildes (Muster). Es geht nun darum, die Muster innerhalb dieser Segmente zu erkennen.

- Ein Muster innerhalb eines Segments wird dadurch (wieder) erkannt, dass es mit einem oder mehreren ***Referenzmustern*** verglichen wird. Dafür ist es nötig, zunächst eine Menge von Referenzmustern zur Verfügung zu stellen. Der Rechner muss also die Referenzmuster (in diesem Fall die Buchstaben A bis Z, die Ziffern 0 bis 9 und die Umlaute Ä, Ö und Ü) „lernen". Das Lernen der Referenzmuster kann z.B. über besonders ausgewählte Fotos von Fahrzeugen erfolgen. Die Referenzmuster werden auch einfach als ***Referenz*** bezeichnet.
- Wurden bereits genügend Referenzen gelernt, so können die Muster der Segmente eines Ausgangsbildes erkannt werden.

Es ergibt sich folgender Ablauf:



2.3 Grundlagen[2]

2.3.1 Applikation und formale Beschreibung

Die Applikation wurde mit Delphi Professional 5 erstellt. Zur Vereinfachung werden deshalb einige Teile der beschriebenen Algorithmen auch in einer Delphi-ähnlichen

[2] zu den Grundlagen der Bildverarbeitung s. z.B. [JAEHN2002], [JAEHN2002a], [GONZ2002] oder [PRATT2001]

Syntax beschrieben. Da Delphi Objektorientierte Programmierung voll unterstützt, werden die entsprechenden Methoden auch eingesetzt, soweit dies sinnvoll ist.

2.3.2 Technische Vorbemerkungen zu Bitmaps

Das für die Mustererkennung nützlichste Graphikobjekt innerhalb der Windows-Umgebung ist das ***Bitmap***, da hierfür viele benötigte Funktionen zur Verfügung stehen, wie farbige Darstellung, direkter Zugriff auf die Pixel, Ausschnitte herauskopieren bzw. verändern, Zwischenablage, Vergrößern/Verkleinern, Rotation, Speichern, Laden etc. Aus diesem Grund wird das Ausgangsbild, sofern es nicht schon als Bitmap vorliegt, in jedem Fall in ein Bitmap umgewandelt. Ein Bitmap ist immer rechteckig, hat eine Breite von w (width) und eine Höhe von h (height). Diese w x h Matrix beinhaltet die **farbigen** Bildpunkte (Pixel). Diese Matrix kann man in dem üblichen Bildschirm-Koordinatensystem darstellen, wobei der Koordinatenursprung (0,0) in der linken obere Ecke liegt und die X-Achse waagerecht nach rechts und die Y-Achse senkrecht nach unten aufgetragen werden. Somit hat die rechte untere Ecke des Bitmaps die Koordinaten (width-1, height-1). Bezeichnet man das Bitmap mit B, so kann man mit der Eigenschaft B(x,y) jeden Bildpunkt des Bitmaps adressieren, um ihn zu lesen, zu verändern oder zu setzen.

Natürlich ist dabei nicht nur das Ausgangsbild ein Bitmap, sondern auch die Segmente sind als Ausschnitte des Ausgangsbildes wiederum Bitmaps.

Zur späteren Erkennung der „Reihenfolge“ der erkannten Muster ist es notwendig, die Lage der Segmente innerhalb des Ausgangsbildes zu kennen. Da die Segmente wiederum Rechtecke sind, benötigt man nun die Koordinaten der linken oberen (top, left) Ecke des Segments in Bezug auf das Koordinatensystem des Ausgangsbildes. Das Segment selber hat eine Breite (width) und Höhe (height), so dass sich die rechte untere (right, bottom) Ecke des Segments in Bezug auf das Koordinatensystems des Ausgangsbildes zu (right:= top + width, bottom:= top + height) ergibt.
Diese Technik der Koordinatenumrechnung (Screen <-> Client) wird u.a. in Windows ausgiebig genutzt. Sie ist z. B. in [**DELPHI5**] beschrieben.

Ein Segment im Sinne der Mustererkennung ist somit zweierlei:

(1) Ein **rechteckiger** Ausschnitt der Breite (width) und Höhe (height) mit den Koordinaten (left, top, left+width-1, top+height-1) = (left, top, right, bottom) bezogen auf das „umgebende“ Ausgangsbild.

(2) Das Segment umschließt ein Bitmap der Breite (width) und Höhe (height). Dessen Bildpunkte (Pixels) beginnen mit den „lokalen“ Koordinaten (0,0) in der linken oberen Ecke und können mit der Eigenschaft B(x,y) adressiert werden.

Bitmaps sind grundsätzlich farbig. Die Farbe eines Bildpunktes wird unter Windows eindeutig über einen 24-bit Integer festgelegt. Dabei repräsentieren jeweils 8-bit (= 1 Byte) die Farben blau (b), grün (g) und rot in **jeweils** 256 Intensitäten bzw. Helligkeiten $(r, g, b \in (0,1,...,255)$. Dies ist das RGB-Format[3]. Dabei ist $FF0000 ein reines (dunkles) blau, $00FF00 ist ein reines (dunkles) grün und $0000FF ist ein reines (dunkles) rot – jeweils in voller Intensität. Sind alle 3 Farbintensitäten gleich, so wird daraus ein Grauwert - $000000 ist reines schwarz, $808080 ist ein mittleres Grau und $FFFFFF ist reines weiß. Folglich beschreibt $c_1 = 256^2 b + 256 g + r$ den 24-bit Integer einer beliebigen Farbe mit den Intensitäten b, g und r. In den meisten Programmiersprachen, so auch in Delphi, existiert dafür eine Funktion ***RGB(r, g ,b),*** die die gewünschte Farbe aus den 3 vorgegebenen Intensitäten errechnet.

2.4 Vorverarbeitung

Zur Erkennung von Autokennzeichen genügt eine monochrome Darstellung mit schwarzen und weißen Bildpunkten. Aus diesem Grund wird, sofern nötig, das (farbige) Ausgangsbild über eine Graustufendarstellung in eine monochrome Darstellung umgewandelt. Weiterhin kann das Ausgangsbild unter unterschiedlichen Lichtverhältnissen (hell/dunkel; Tag/Nacht; Sonne/Regen etc.) entstanden sein. Es kann verschmutzt oder verdreht sein. Daraus wird ersichtlich, dass vor dem eigentlichen Lern- bzw. Erkennungsvorgang eine mehr oder weniger aufwändige Vorverarbeitung nötig ist. Diese Schritte sollen im Folgenden beschrieben werden.

[3] Leider weicht die Bezeichnung RGB von der internen Darstellung des 24-bit Integer mit „BGR“ ab; der Begriff hat sich aber so etabliert und soll selbstverständlich auch hier so weiter verwendet werden.

2.4.1 Graudarstellung/Monochrom

2.4.1.1 Einleitung

Farbige Bilder benötigen bei den heutigen Auflösungen selbst einfacher Digitalkameras (> 2 Mio. Bildpunkte mit i.d.R. 24 bit Farbtiefe) riesigen Speicherplatz, wobei zudem die Menge der zu verarbeitenden Daten die Laufzeiten der Algorithmen negativ beeinflusst. Es ist also zweckmäßig, die Ausgangsbilder und die Segmente bzw. Referenzen über einen Zwischenschritt mit 256 Grautönen in eine monochrome Darstellung umwandeln. Dadurch gehen zwar einige Informationen verloren, die Verfahren werden dann aber deutlich einfacher und schneller. In dieser Arbeit wird ausschließlich von monochromen Mustern (Ausgangsbild, Segmente und Referenzen) ausgegangen.

2.4.1.2 Grauwertermittlung

Soll nun für eine beliebige Farbe c_1 der zugehöriger Grauwert c_2 ermittelt werden, so bieten sich zwei Verfahren an:

(1) Man ermittelt den numerisch nächst kleineren oder nächst größeren ganzzahligen Wert zu c_1, wo alle Intensitäten gleich groß sind. Daraus folgt aber: $c_2 = 256^2 c + 256c + c = (256^2 + 256 + 1)c = 65793c$ und damit $c = [65739 / c_1]$. Für c_2 ergibt sich dann $c_2 = RGB(c,c,c)$.

(2) Man ermittelt den ganzzahlig gerundeten Mittelwert der 3 Intensitäten $c = [(b + g + r)/3]$ und berechnet daraus den Grauwert $c_2 = RGB(c,c,c)$

Wird dies für jeden Bildpunkt des Ausgangsbildes durchgeführt, erhält man das Bild in 256 Graustufen.

Beide Verfahren liefern bei gleichem Aufwand unterschiedliche Ergebnisse. Das erste Verfahren berücksichtigt mehr die Gesamtfarbe, das zweite mehr die einzelnen Farbintensitäten. In der Applikation sind beide Optionen einstellbar.

2.4.1.3 Schwellwertverfahren

Die Umwandlung in ein monochromes Bild mittels Schwellwertverfahren ist jetzt recht einfach. Dazu wird ein Schwellgrauwert $c_s \in \{0,1,...,255\}$ gewählt. Alle Bildpunkte dunkler RGB(c_s, c_s, c_s) werden schwarz, die anderen weiß (vgl. auch [**FREY**]).

Fraglich ist nun, welchen Wert c_s annehmen soll. Dazu bieten sich verschiedene Möglichkeiten an:

(1) Alle Bildpunkte, die dunkler als RGB(128,128,128) sind, werden schwarz, die anderen werden weiß dargestellt.

(2) Es wird über das komplette Ausgangsbild der Mittelwert aller Grautöne bestimmt. Dieser Grauton ist dann der Schwellgrauwert c_s.

(3) Es wird ein rechteckiger Ausschnitt des Originalbildes gewählt. Dessen Mittelwert aller im Ausschnitt liegender Grauwerte ist dann der Schwellgrauwert c_s.

(4) Das menschliche Auge kann nah beieinander liegende Grauwerte nur schwer auseinander halten. Projiziert man die 256 möglichen Grautöne auf das Intervall [0,1], so werden die Grautöne 0.0 bis 0.1 z.B. genauso unterschiedlich wahrgenommen, wie die Grautöne von 0.5 bis 0.55 [**PLA1986** Seite 362].

Zu (1)

Diese Methode berücksichtigt in keiner Weise unterschiedliche Lichtverhältnisse usw. und führt nur bei „Idealbildern“ zu brauchbaren Ergebnissen.

Zu (2)

Im Gegensatz zu (1) werden hier zwar Bildinformationen genutzt, aber in ihrer Gesamtheit und nicht auf den Ausschnitt des Autokennzeichens beschränkt. Dies kann dazu führen, dass sowohl die schwarzen als auch die weißen Anteile des Autokennzeichens unter- oder oberhalb des Schwellwertes liegen. Damit würde der Bereich des Autokennzeichen komplett zu schwarz oder weiß konvertiert.

Zu (3)

Abhängig von den Lichtverhältnissen am Ort der Aufnahme, ist schwarz und weiß ein subjektiver Begriff. Dazu ein Beispiel:

Bild 1 Helles Foto

Bei diesem in der hellen Sonne aufgenommen Foto hat schwarz im Kennzeichen einen Grauwert von ca. 60 und weiß von ca. 215. Der Kontrast ist mit 215 – 60 = 155 also recht hoch.

Bei diesem Foto, ebenfalls bei Sonnenlicht aufgenommen, liegt das Nummernschild aber im Schatten.

Bild 2 Dunkles Foto

Das schwarze K hat einen Grauwert von ca. 6, weiß ca. 60. Dem optischen Wahrnehmungsvermögen widerspricht, dass „Weiß“ im zweiten Bild annähernd den gleichen Grauton hat, wie „Schwarz“ im ersten Bild. Für das erste Bild würde ein Schwellgrauwert von 128 ausreichen, das Bild nach monochrom zu wandeln. Für Bild 2 würde 128 zu einem komplett schwarzen Bild führen.

Wenn man ungefähr weiß, wo sich das Kennzeichen befindet, kann man für dieses Rechteck das gewogene Mittel an Grautönen ermitteln und diesen Wert als Schwellgrauwert nehmen. In der Praxis wird man davon ausgehen können, dass die Kamera immer an einem festen Punkt installiert ist. Somit könnte man im Fokusbereich der Kamera eine quadratische Tafel aus dem Material installieren, aus denen üblicherweise Autokennzeichen gefertigt werden. Die eine Hälfte ist schwarz, die andere Hälfe ist weiß. Wenn man dafür sorgt, dass diese „Referenztafel“ den gleichen Lichtverhältnissen ausgesetzt ist, wie das zu fotografierende Autokennzeichen, hat man einen Referenzwert für Schwarz und einen für Weiß. Dessen Mittelwert ist dann der Schwellgrauwert c_s.

Zu (4)

Will man einen beliebigen Grauwert $c \in \{0,1,...,255\}$ aus 256 Grautönen **logarithmisch** auf zwei Farben $c_2 \in \{0,1\}$ abbilden, so kann man dies für c_2 nach kaufmännischer Rundung folgendermaßen wählen:

$$c_2 = \left[2^{\frac{c}{255}} \right] \text{-1}$$

Fraglich ist nun, bis zu welchem Wert von c der Parameter c_2 gerade noch auf 0 bzw. ab wann c_2 auf 1 gerundet wird. Dieses c (ganzzahlig gerundet) ist dann der Schwellwert c_s.

Es ergibt sich:

$$2^{\frac{c_s}{255}} - 1 = 0.5$$

$$2^{\frac{c_s}{255}} = 1.5$$

$$\frac{c_s}{255} = \log_2(1.5)$$

$$c_s = 255 \log_2 1.5$$

$$c_s \approx 255 \cdot 0{,}585$$

$$c_s \approx 149.17 \text{ (gerundet 149)}$$

Das heißt, alle Grauwerte c dunkler als c_s werden hier auf „schwarz“, die anderen werden auf „weiß“ gesetzt.

2.4.1.4 Kantenerkennung

Der Schwellwert c_s gilt für das gesamte Ausgangsbild und berücksichtigt nicht, dass es innerhalb des Bildes stark unterschiedliche Kontraste geben kann. Zur Behebung dieser Schwierigkeiten können Kantenerkennungsverfahren eingesetzt werden, die die Abgrenzung der schwarzen Kennzeichen auf weißem Grund ausnutzen. Mögliche Methoden dazu sind die Kantenerkennung nach Sobel oder andere [**JENSCH**], [**LUEN**] oder [**SUND1999**] bzw. die Skelettierung nach Zhang/Suen [**MART1997** Seite 19 ff]

Beispiel:

Bild 3 Ausgangsbild

Bild 4 Ausgangsbild nach der Kantenerkennung nach Sobel

Bei meinen praktischen Versuchen hat sich gezeigt, dass diese Verfahren sehr anfällig auf „Rauschen“ reagieren, da Kontrastunterschiede natürlich nicht nur an den Ziffern/Buchstaben auftreten, sondern auch im gesamten Bild bzw. auch an den Rändern.

2.4.1.5 Vereinfachung bei monochromen Bildern

Liegt das Ausgangsbild monochrom vor, direkt oder nach einer Umwandlung, sind auch die weiteren Graphikobjekte monochrom. Somit kann ein Bildpunkt nur schwarz (gesetzt) oder weiß (nicht gesetzt) sein. Für die weitere algorithmische Verarbeitung ist es sinnvoll, die bisherigen Farbkonstanten für weiß ($FFFFFF) und schwarz ($000000) zu verlassen und für einen gesetzten Bildpunkt eine 1 und für einen nicht gesetzten Bildpunkt eine 0 zu wählen. Somit wird die o.g. Eigenschaft B(x,y) durch die boolesche Eigenschaft B(x,y) überschrieben:

$$B(x,y) = \begin{cases} 1, \textit{wenn der Bildpunkt mit den Koordinaten } (x,y) \textit{ gesetzt ist} \\ 0, \textit{sonst} \end{cases}$$

2.4.2 Segmentieren

In den seltensten Fällen wird das gesamte Bild dem zu erkennenden Muster entsprechen. Normalerweise werden die Muster aus einem oder mehreren Teilen des Bildes bestehen. Für die Autokennzeichen bedeutet dies, dass die einzelnen Buchstaben und Ziffern „irgendwo“ auf dem Foto sein können. Da das Ausgangsbild monochrom ist, sind die Buchstaben und Ziffern schwarz.

Eine Segmentierung könnte darin bestehen, um jedes Zeichen einen rechteckigen Rahmen zu ziehen, der das Muster umschließt. Manuell würde man sicherlich das Bild segmentieren, indem man mit der Maus auf einem weißen Punkt oben links des zu segmentierenden Bereichs klickt und dann mit der Maus einen rechteckigen Rahmen um den gewünschten Bereich zieht. Sofern alle Punkte auf diesem Rahmen weiß und mind. ein Punkt innerhalb des Rahmens schwarz ist, hat man ein Segment gefunden. Dieses Verfahren wiederholt man, bis alle Muster umrahmt sind. Damit der Algorithmus terminiert und „notfalls“ das ganze Bild als ein Segment erkannt wird, setzt man alle äußeren Bildpunkte auf weiß. Dies kann über zwei Verfahren geschehen:

- Direktes Zeichnen von weißen Linien in das Bild. Dadurch gehen aber evtl. Informationen verloren.
- Einfügen von zwei „Leerzeilen“ oben und unten und zwei „Leerspalten“ links und rechts. Die Bildpunkte dieser „Leerzeilen“ bleiben weiß.

Algorithmisch kann man analog vorgehen:

Man beginnt in der linken oberen Ecke des Bitmaps B(0,0) und sucht zeilenweise von links nach rechts und von oben nach unten, nach einem weißen Bildpunkt, der rechts und unterhalb ebenfalls weiße Nachbarn und rechts unten einen schwarzen Nachbarn hat. Dies ist der Startpunkt (x_0,y_0) für die weitere Betrachtung. Es gilt also:

$B(x_0, y_0) = 0$
$B(x_0 + 1, y_0) = 0$
$B(x_0 + 1, y_0 + 1) = 1$
$B(x_0, y_0 + 1) = 0$

Denkt man sich das Bild in Quadrate mit 3 x 3 Bildpunkten zerlegt, stellt sich diese Situation in einem der 3 x 3 Quadrate so dar:

	x_0	x_0+1	x_0+2
y_0	0	0	b
y_0+1	0	1	b
y_0+2	b	b	b

Die Felder mit b können beliebig schwarz oder weiß sein.

Nun „zieht“ man einen rechteckigen Rahmen ausgehend von (x_0, y_0) nach rechts und unten mit der Breite 3 und Höhe 3. Dieses Rechteck hat somit die Koordinaten $(x_0, y_0, x_0 + 2, y_0 + 2) = (x_0, y_0, x_1, y_1)$ mit $x_1 = x_0 + 2$ und $y_1 = y_0 + 2$

Für jede der 4 Kanten des Rechtecks (oben, rechts, unten und links) wird geprüft, ob sie nur weiße Bildpunkte enthält.

Dabei müssen folgende Fälle unterschieden werden:

a) Wenn alle 4 Kanten nur aus weißen Bildpunkte bestehen, hat man ein Segment gefunden. Im einfachsten Fall ist dies das Rechteck um den Punkt (x_0+1,y_0+1), der ja definitionsgemäß schwarz ist, sonst wäre (x_0,y_0) nicht Startpunkt geworden.
b) Findet man auf der oberen Kante mindestens einen schwarzen Bildpunkt, so wird das Rechteck nach oben um eine Zeile vergrößert; d.h. y_0:=y_0-1
c) Findet man auf der rechten Kante mindestens einen schwarzen Bildpunkt, so wird das Rechteck nach rechts um eine Spalte vergrößert; d.h. x_0:=x_0+1

d) Für die untere und linke Kante wird analog verfahren.

Bei a) hat man ein Segment gefunden. Die Koordinaten des Rechtecks werden in einer Liste gespeichert, und man sucht einen neuen Startpunkt (x_0,y_0), indem man vom vorigen Startpunkt aus weiter zeilenweise von links nach rechts und von oben nach unten sucht, bis man das ganze Bild untersucht hat.
Bei b) – d) wiederholt sich das Verfahren mit den neuen Koordinaten (x_0,y_0,x_1,y_1).
Da man vorher um das Bild einen weißen Rahmen gesetzt hat, terminiert dieser Algorithmus nach endlich vielen Schritten (Komplexität $O((B.w \cdot B.h)^2)$).

Ein alternatives Verfahren zu dieser „Gummiband"-Methode [**FREY**] ist, ausgehend von den Startpunkten (x_0,y_0), mit Hilfe des Freeman-Ketten-Algorithmus [**MART1997** Seite 20 ff] (Komplexität $O((B.w \cdot B.h)^2)$) ein Muster zu umfahren, um daraus die Umrisse zu erhalten. Aus den größten und kleinsten X- bzw. Y-Koordinaten kann man dann das „umhüllende" Segmentrechteck ermitteln.
Beiden Verfahren ist gemeinsam, dass Segmente lediglich „umfahren" werden – es gibt keinerlei Information, über das „Innere" des Segments, in dem weitere Segmente enthalten sein können.
Ein Verfahren, das diesen Nachteil vermeidet und auch noch deutlich schnellere Ergebnisse liefert, wird bei [**DIET2003**] beschrieben. Dabei wird davon ausgegangen, dass jedes Segment aus zusammenhängenden bzw. benachbarten Punkten besteht. Man sucht also, analog zum Operations-Research, nach Zusammenhangskomponenten [**NEU1993**].

Dazu wird im Bild von der linken oberen Ecke aus zeilenweise nach dem ersten schwarzen Bildpunkt gesucht. Dieser Bildpunkt bildet den Ausgang für ein Segment. Nun werden die max. 8 „schwarzen" Nachbarpunkte dieses Bildpunktes gesucht und dem Segment zugeordnet. Für diese „schwarzen" Nachbarn werden wiederum (rekursiv oder über Listen) die max. 8 „schwarzen" Nachbarn gesucht usw.; dabei muss allerdings Vorsorge getroffen werden, damit keine Punkte mehrfach gezählt werden. Lassen sich keine weiteren schwarzen Nachbarn mehr finden, bilden die bisher gefundenen (zusammenhängenden) Punkte ein Segment, das in einer Liste gespei-

chert wird. Aus den größten und kleinsten X- bzw. Y-Koordinaten kann man das „umhüllende“ Segmentrechteck ermitteln.

Nun wird im „restlichen“ Bild nach ggf. noch vorhandenen schwarzen Bildpunkten gesucht und o.g. Algorithmus erneut angewandt. Am Ende des Verfahrens hat man eine Liste von Segmenten.

Wie man sieht, ist dieses Verfahren proportional zu der Anzahl Bildpunkte; seine Komplexität könnte man durch O(B.w·B.h) beschreiben.

2.4.3 Filter

Der Segmentierungsalgorithmus (oder einfach auch Segmentierer genannt) liefert alle durch ein Rechteck umrahmbare Bildausschnitte. Dies können aber nicht nur die Buchstaben und Ziffern des Autokennzeichens sein, sondern auch Schmutzflecken, dunkle Schrauben etc.

Bild 5 Beispiel für ein Kennzeichen mit sichtbaren Schrauben

Bild 6 Die Segmentierung erkennt auch die Schrauben

Häufig ist die Fläche solcher unerwünschter Segmente sehr klein (im Grenzfall ist nur ein Punkt enthalten). Definiert man nun eine Mindestfläche für Segmente (Breite x Höhe), so können alle Segmente mit einer Fläche kleiner dieser Mindestfläche aus der Liste der Segmente gestrichen und deren Punkte im Ausgangsbild auf weiß gesetzt werden. Damit wird das Ausgangsbild um Schmutz, Schrauben u.ä. gesäubert.
Häufig befindet sich, wie im obigen Beispiel, die Befestigungsschraube zwischen zwei Ziffern des Kennzeichens. Die Lage dieser Schraube kann so ungünstig sein, dass der Segmentierer bei der Gummiband-Methode die beiden Ziffern nicht als getrennte Segmente unterscheiden kann, da die Schraube einen rechteckigen Rahmen

um die einzelnen Ziffern verhindert. Die Schraube selber wird aber erkannt, genauso wie die beiden Ziffern incl. der Schraube. Wird o.g. Mindestflächenfilter aktiviert, wird die Schraube aus dem Ausgangsbild entfernt. Ein Wiederholen des Segmentierers wird dann ggf. beide Ziffern sauber unterscheiden können.

Bild 7 Die Schraube wurde aus dem Kennzeichen entfernt

2.4.4 Teilen

Trotzdem kann es vorkommen, dass der „Fliegendreck“ nicht komplett entfernt werden kann, da um den “Fliegendreck“ kein Rechteck gezeichnet werden kann oder mehrere Zeichen als zusammenhängend erkannt werden. Dann werden ggf. mehrere Zeichen gemeinsam umrahmt, wie in dem folgenden Beispiel die 4,5 und die 8.

Bild 8 Zeichen werden gemeinsam umrahmt

Da zwischen den einzelnen Ziffern spaltenweise nur sehr wenige Pixel schwarz sind, können Spalten mit einer geringen Pixelanzahl als Zwischenräume interpretiert werden. Das gesamte Rechteck wird dann in mehrere kleinere Rechtecke aufgeteilt.

Bild 9 Zwischenräume werden erkannt

2.4.5 Neutralisation von 1-Pixel-Fehlern

Durch Vergrößerung, Verkleinerung, Unschärfen etc. entstehen an den Kanten der Buchstaben und Ziffern häufig sog. 1-Pixel-Fehler. Zur Neutralisation dieser Fehler können Filterschablonen in Form einer 3x3-Matrix über jeden Bildpunkt gelegt werden, wobei der zu untersuchende Bildpunkt vom „mittleren Element“ der Matrix überdeckt wird. Stimmt die Umgebung des zu untersuchenden Punktes in allen Punkten mit der Filterschablone überein, so wird der Bildpunkt invertiert.

Dabei haben sich folgende 8 Filterschablonen als sinnvoll erwiesen [**MART1997**]:

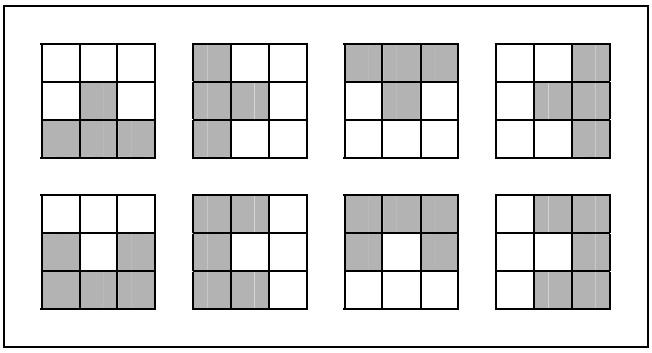


Abbildung 1 Filterschablonen zur Neutralisation von 1-Pixel Fehlern

Beispiel (Buchstabe I):

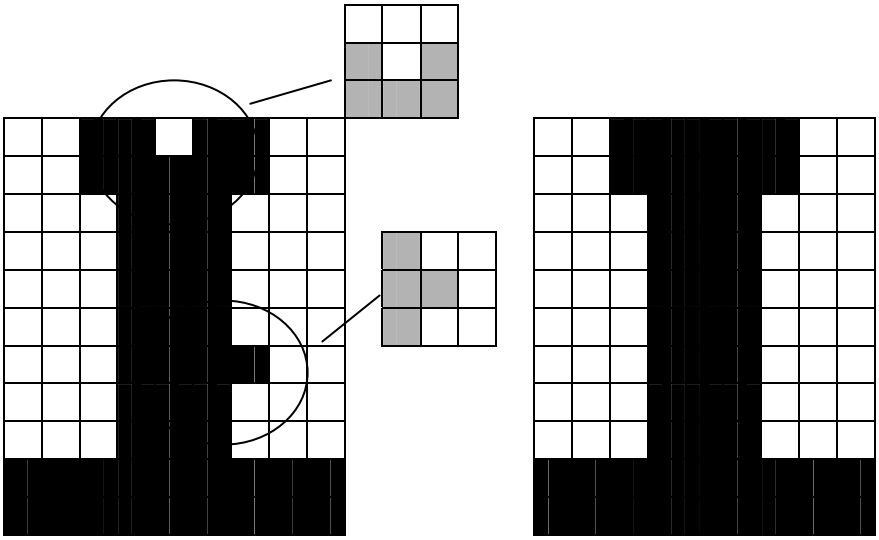


Abbildung 2 1-Pixel-Fehler Neutralisation am Buchstaben I

2.4.6 Zeichengröße

Bislang erkennt der Segmentierer auch die TÜV-Stempel, Zulassungsplaketten etc. Da die Buchstaben/Ziffern eines Autokennzeichens annähernd gleich groß sind, lässt sich eine Zeichengröße (Breite, Höhe) definieren. Liegen die gefundenen Segmente unter Beachtung einer Toleranz ober- oder unterhalb dieser Zeichengröße, so werden diese Segmente ebenfalls aus der Liste gestrichen.

Bild 10 Segmente außerhalb der Zeichengröße werden eliminiert

2.4.7 Umlaute

Der Segmentierer erkennt bisher noch keine Umlaute, sondern findet die Vokale und die „dazugehörigen“ Punkte als separate Segmente, sofern die Mindestgröße der Segmente („Fliegendreck“) nicht zu groß eingestellt wurde und damit die Punkte gelöscht wurden:

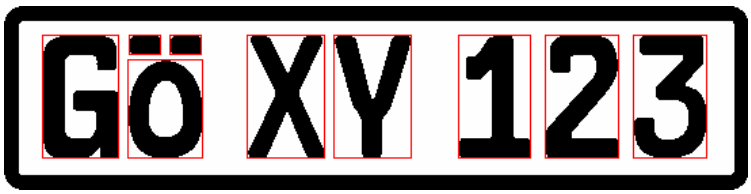


Bild 11 Punkte über den Konsonanten werden als eigene Segmente erkannt

Die Vokale der Umlaute sind deutlich niedriger als ihre „normalen Kollegen“. Die Punkte und die Vokale übereinander haben aber wieder die Größe eines normalen Zeichen. Außerdem hat die linke obere Ecke des Punktes annähernd die gleiche x-Koordinate wie die linke obere Ecke des Vokals; die rechte obere Ecke des rechten Punktes hat annähernd die gleiche x-Koordinate, wie die rechte obere Ecke des Vokals.

Daraus lässt sich folgender Algorithmus entwickeln:

Die gefundenen Segmente werden paarweise verglichen. Genügen beide Segmente o.g. Bedingung, so werden diese beiden Segmente markiert und um beide Segmente ein umhüllendes Segment gelegt. Das umhüllende Segment wird der Liste der Segmente zugefügt. Am Ende des paarweisen Vergleichs werden die markierten Segmente gelöscht. Die umhüllenden Segmente enthalten jetzt die Vokale mit den Punkten.

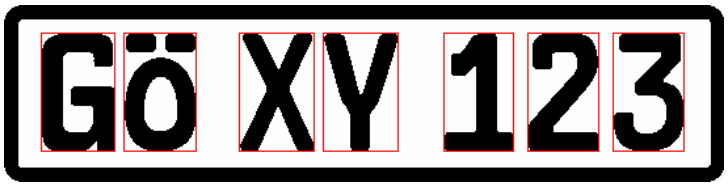


Bild 12 Erkennung der Umlaute

2.4.8 Ausrichtung

Bei einer ungünstigen Kamera- und/oder Objektposition kann es passieren, dass das Autokennzeichen auf dem Ausgangsbild nicht waagerecht ist.

Bild 13 Autokennzeichen ist nicht waagerecht

Wäre das Kennzeichen waagerecht, so würden die y-Koordinaten der linken unteren Ecken der Segmente annähernd auf einer waagerechten Geraden liegen.

Bild 14 Segmente liegen nicht auf einer Geraden

Ist dies nicht der Fall, bietet sich an, über die Koordinaten der linken unteren Ecken der gefundenen Segmente eine **lineare Regression** durchzuführen. Aus der Steigung der gefundenen Regressionsgeraden lässt sich der notwendige Rotationswinkel ermitteln, um dessen **negativen** Wert das Ausgangsbild gedreht wird.
Nach der erneuten Segmentierung liegen dann die Koordinaten der linken unteren Ecke „optimal“ auf der dann waagerechten Regressionsgeraden.

Bild 15 Bild wurde gedreht

2.4.9 Iteration

Werden diese Stufen (Phasen) ggf. mehrfach durchlaufen, so lassen sich die Segmente iterativ bis zu einer gewünschten Qualitätsstufe verbessern.

Am Ende des Verfahrens liegt dann das Bild des Autokennzeichens zerlegt in seine relevanten Segmente (in diesem Fall Buchstaben/Ziffern) vor. Diese Segmente können nun zum Lernen bzw. zur Erkennungslogik weitergeleitet werden. Zunächst muss aber geklärt werden, woran die Segmente „erkannt“ werden können.

2.5 Merkmalscodierung

2.5.1 Einleitung

Die Segmente werden bis jetzt lediglich beschrieben durch die Koordinaten der linken oberen und rechten unteren Ecke, der Breite und Höhe, sowie dem Muster (ein Bitmap) innerhalb des Segments. Sowohl zum Erkennen als auch zum Lernen genügt diese Information natürlich nicht. Für diese Segment-Bitmaps müssen nun Merkmale gefunden werden, anhand derer eine Erkennung möglich ist. Anschließend müssen diese Merkmale codiert und effizient gespeichert werden.

2.5.2 Merkmale

2.5.2.1 Vorbemerkung

Die Merkmale werden alle am Beispiel des Bitmaps „LEO“ dargestellt [**BECK1997**, Seite 19 ff]. Wie man sieht, ist das Bitmap monochrom und hat eine Breite von 15 und eine Höhe von 5 Pixeln.

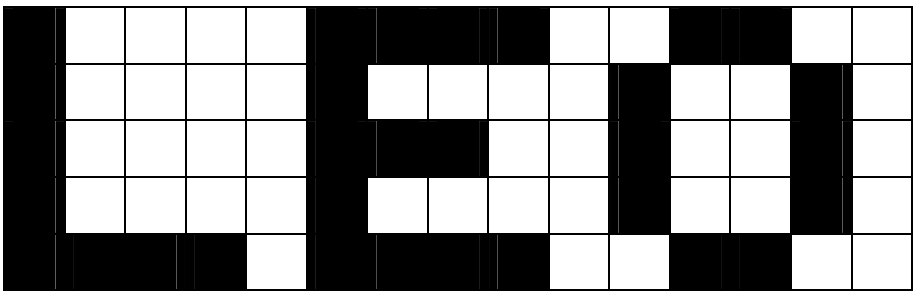


Bild 16 Schriftzug „LEO“ als Basis für die weiteren Überlegungen

Obwohl die Koordinaten eines Bitmap in der oberen linken Ecke (0,0) beginnen und x nach rechts und y nach unten größer wird, soll bei den weiteren Betrachtungen davon ausgegangen werden, dass das Muster im 1. Quadranten eines kartesischen Koordinatensystem abgebildet ist. Somit ist die untere Ecke (0,0) und x wird nach rechts und y nach oben größer. Die dafür notwendige Umrechnung der y-Koordinaten ist trivial: y:= B.h – y – 1. Wie bisher wird die Breite der Graphik mit w (width) und die Höhe mit h (height) bezeichnet. Somit hat die Graphik w·h Bildpunkte.

2.5.2.2 SkyLineX

Bei dem Merkmal SkyLineX werden in einem Vektor der Länge w je Spalte (x-Koordinate) der größte y-Wert der gesetzten Bildpunkte gespeichert.

Definition:

SkyLineX(y) = $(\max y; B(x, y) = 1); x = 0,1,..., B.w - 1$

Beispiel:

Für „Leo“ ergibt sich: SkyLineX = $(5,1,1,1,0,5,5,5,5,0,4,5,5,4,0)$

Grafische Darstellung des Vektors:

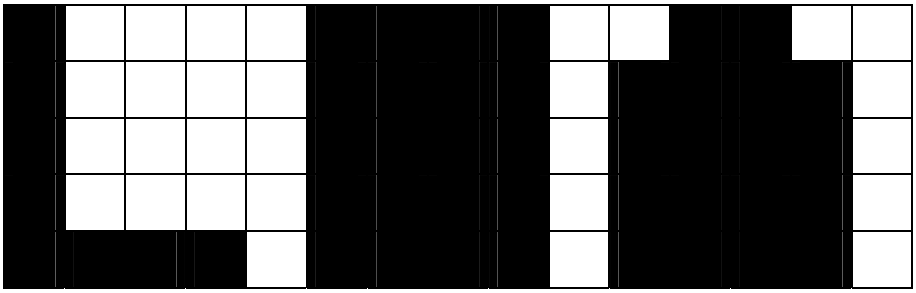


Bild 17 SkyLineX für „LEO“

2.5.2.3 SkyLineY

Bei dem Merkmal SkyLineY wird in einem Vektor der Länge B.h je Zeile (y-Koordinate) der größte x-Wert der gesetzten Bildpunkte gespeichert.

Definition:

SkyLineY(x) = $(\max x; B(x, y) = 1); y = 0,1,..., B.h - 1$

Beispiel:

SkyLineY = $(13,14,14,14,13)$

Graphische Darstellung des Vektors:

Bild 18 SkyLineY für „LEO“

2.5.2.4 X-Projection

Bei der X-Projection werden je Spalte (x-Koordinate) die Anzahl der gesetzten Bildpunkte gezählt.

Definition:

$$\text{XProjection(x)} = \sum_{y=0}^{B.h-1} B(x,y); \quad x = 0,1,...,B.w-1$$

Beispiel:

$$\text{XProjection} = (5,1,1,1,0,5,3,3,2,0,3,2,2,3,0)$$

Graphische Darstellung des Vektors:

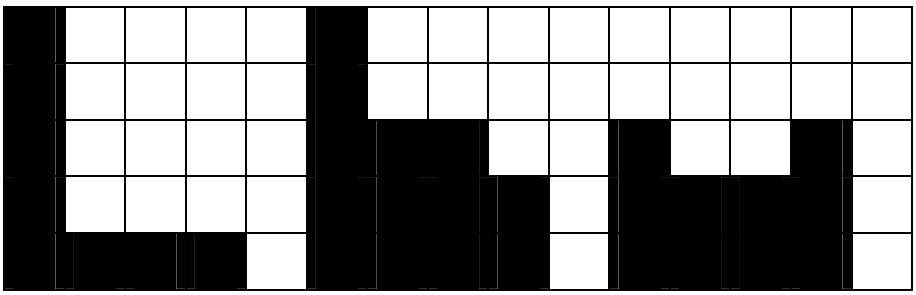


Bild 19 X-Projection für „LEO“

2.5.2.5 Y-Projection

Bei der Y-Projection werden je Zeile (y-Koordinate) die Anzahl der gesetzten Bildpunkte gezählt.

Definition:

$$\text{YProjection(y)} = \sum_{x=0}^{B.w-1} B(x,y); y = 0,1,...,B.h-1$$

Beispiel:

YProjection = $(10,4,6,4,7)$

Graphische Darstellung des Vektors:

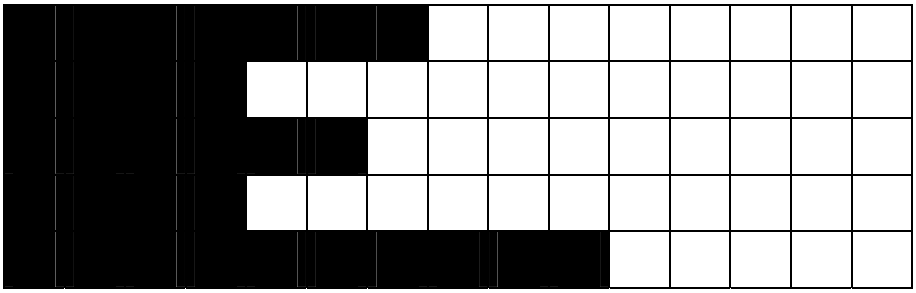


Bild 20 Y-Projection für „LEO“

2.5.2.6 GroundLine

Bei der GroundLine werden je Spalte (x-Koordinate) die **nicht** gesetzten Bildpunkte **unterhalb** des Musters gezählt. Es entspricht also dem Tiefenzug des Musters. Sollte in Spalte x kein Bildpunkt gesetzt sein, so wird die GroundLine für diese Spalte auf die Höhe des Bitmaps gesetzt. Eine Spalte die nur gesetzte Pixel enthält hat den Wert 0.

Definition:

$$GroundLine(x) = \begin{cases} \min(y; B(x,y) = 1) - 1, wenn \sum_{y=0}^{B.h-1} B(x,y) \geq 1 \\ B.h, sonst \end{cases} \quad x = 0,1,...,B.w-1$$

Beispiel:

$GroundLine = (0,0,0,0,5,0,0,0,0,5,1,0,0,1,5)$

Graphische Darstellung des Vektors:

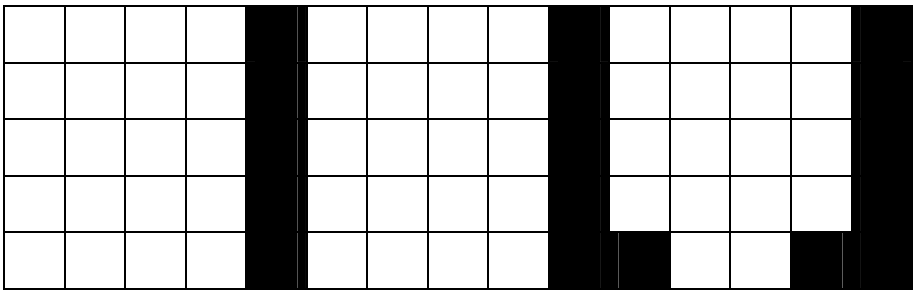


Bild 21 GroundLine für „LEO“

2.5.2.7 Gitter

Jede der bisher genannten Merkmale lassen sich, wie dargelegt, graphisch darstellen (die Abmessungen entsprechen dem zu untersuchenden Bitmap und damit B.w

und B.h). Nun kann man über jede dieser Graphiken ein Gitter legen, das die Graphik in m Zeilen und n Spalten aufteilt. Damit die dadurch entstehenden m·n großen Felder alle gleich groß sind, muss die Höhe der Graphik ganzzahlig durch m und die Breite der Graphik ganzzahlig durch n teilbar sein; also B.w mod n = 0 und B.h mod m = 0. Jedes Feld enthält damit $\frac{B.w}{n} \cdot \frac{B.h}{m}$ Bildpunkte. Nummeriert man diese Felder beginnend bei 0 (z.B. zeilenweise von unten links nach rechts oben) durch und zählt die in jedem Feld gesetzten Bildpunkte, so erhält man einen Vektor $G = (g_0, g_1,....,g_{mn})$ der Länge m·n, mit der Anzahl gesetzter Bildpunkte je Feld.

Beispiel für die XProjection von „LEO“:

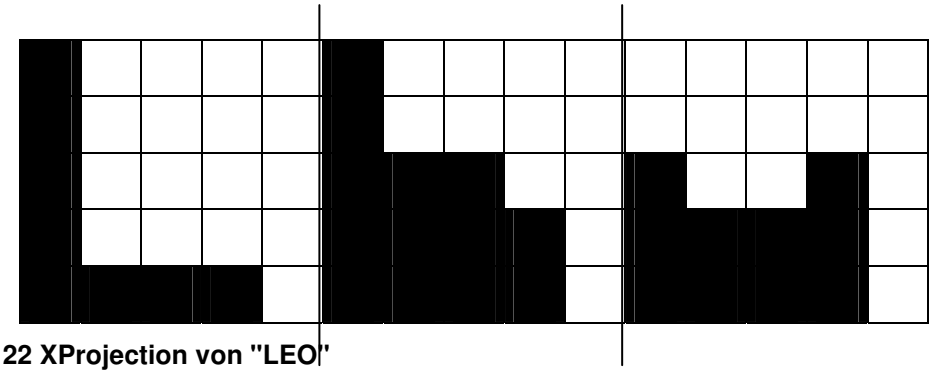


Bild 22 XProjection von "LEO"

Die Breite ist, wie gehabt, B.w = 15 und die Höhe B.h = 5. Hier bietet sich z.B. an m = 1 und n = 3 zu setzen. Die Graphik wird dabei in m·n = 3 Felder mit jeweils (15/3)·(5/1) = 25 Bildpunkte aufgeteilt. Der Vektor G ergibt sich zu G = (8,13,10).
Analog kann auch für die SkyLine, Groundline und das Ausgangsbild verfahren werden.

Das Merkmal Gitter liefert Informationen über die Häufung von Bildpunkten in Teilen der Graphik.

2.5.2.8 Pixelvergleich

Ein „besonderes“ Gitter ist jenes, bei dem die Anzahl Zeilen der Höhe (also m = B.h) und die Anzahl Spalten der Breite (also n = B.h) des Bitmaps entspricht. Die dadurch entstehenden Felder enthalten genau **einen** Bildpunkt, der entweder gesetzt oder nicht gesetzt ist. Somit ist die Anzahl gesetzter Bildpunkt **je Feld** entweder 0 oder 1.

Der (Binär-) Vektor für „LEO“ (beginnend unten links nach oben rechts) ergibt sich damit zu:

G= (1,1,1,1,0,1,1,1,1,0,1,1,1,1,0,
1,0,0,0,0,1,1,1,1,0,1,1,1,1,0,
1,0,0,0,0,1,1,1,0,0,1,0,0,1,0,
1,0,0,0,0,1,0,0,0,0,0,0,0,0,0,
1,0,0,0,1,0,0,0,0,0,0,0,0,0,0)

2.5.2.9 Stabilität

Die Stabilität ist einfach das Verhältnis von Höhe zu Breite des Musters.

Definition

$$S_B = \frac{B.h}{B.w}$$

Beispiel:

„LEO“ hat eine Höhe von 5 und eine Breite von 15 Pixeln. Somit ist

$$S_B = \frac{5}{15} = \frac{1}{3} = 0,\bar{3}$$

2.5.2.10 Schwerpunkt

Zunächst wird die **Gesamtanzahl** gesetzter Punkte n ermittelt. Das Muster wird dann zuerst **vertikal** in zwei Teile zerlegt. Ein vorzugebender Faktor t bestimmt in %, in welcher Spalte (column) $c = tB.w$ die Teilung stattfindet. Nun werden im linken Teil des Musters die gesetzten Punkten n_c gezählt; die Anzahl der Punkte im rechten Teil ergibt sich als Differenz zu n. Ist die Anzahl der Punkte im linken Teil kleiner als im rechten Teil, so wird der **vertikale** Schwerpunkt c_c auf 0 gesetzt. Ist die Anzahl der Punkte in beiden Teilen gleich, so ist c_c=1; sind im rechten Teil mehr Punkte gesetzt, so wird c_c=2.

Analog wird bei einer **horizontalen** Teilung in Reihe (row) $r = tB.h$ verfahren. Die Anzahl Punkte im oberen Teil wird mit n_r bezeichnet; der **horizontale** Schwerpunkt folglich mit c_r. Aus den beiden „lokalen“ Schwerpunkten wird dann der Gesamtschwerpunkt C_B des Bitmap ermittelt: $C_B = 3c_c + c_r$. Für C_B gilt somit $0 \le C_B \le 8$

Definition:

$$n = \sum_{x=0}^{B.w-1} \sum_{y=0}^{B.h-1} B(x,y)$$

$$c = \lfloor tB.w \rceil$$

$$n_c = \sum_{x=0}^{c} \sum_{y=0}^{B.h-1} B(x,y)$$

$$c_c = 1 + \mathrm{sgn}(2n_c - n)$$

$$r = [tB.h]$$

$$n_r = \sum_{x=0}^{B.w-1} \sum_{y=0}^{r} B(x,y)$$

$$c_r = 1 + \mathrm{sgn}(2n_r - n)$$

$$C_B = 3c_c + c_r$$

Beispiel für „LEO“:

Das Muster hat n = 31 gesetzte Bildpunkte. Für t = 0,5 = 50% ist c = 8 und im linken Teil gibt es $n_c = 18$ gesetzte Pixel. Damit hat das Muster im rechten Teil 31-18 = 13 gesetzte Pixel. Der horizontale Schwerpunkt liegt also im linken Teil und damit ist $c_c = 0$.

Für t = 0,5 = 50% ist r = 3. Im unteren Teil sind damit 20 Pixel gesetzt (im oberen Teil folglich 31-20 = 11). Damit liegt der Schwerpunkt demnach im unteren Teil und damit ist auch $c_r = 0$. Der Gesamtschwerpunkt C_B ist damit auch 0.

Weitere Informationen über Merkmale finden sich auch bei **[DIET2003a]**, wobei diese Merkmale aber teilweise für meine Aufgabe nicht relevant waren (z.B. mittlerer Grauwert).

2.5.3 Normierung

Wie man sieht, hängt die Länge der Vektoren der Merkmalscodierungen von den Abmessungen des Bitmaps ab. So hat z.B. der SkyLineX-Vektor eine Länge von B.w, die YProjection eine Länge von B.h und der Schwerpunkt eine konstante Länge von 1. Nach der Segmentierung werden die Segmente i.d.R. alle eine unterschiedli-

che Größe haben. Um später die Segmente des Ausgangsbildes zur Erkennung mit den Referenzen vergleichen zu können, müssen die Segmente und die Referenzen alle eine einheitliche Größe haben, da sonst keine objektiven Vergleiche möglich sind. Aus diesem Grund müssen alle Segmente und Referenzen auf **eine** Größe normiert werden. Dabei muss, um Verzerrungen zu vermeiden, das bisherige Breiten-Höhen-Verhältnis beibehalten werden. Nach der Normierung haben alle gleichen Merkmalscodierungen aller Segmente und Referenzen die gleiche Länge und können verglichen werden. Bei Autokennzeichen kann diese Normierung z.B. auf 32 x 32 gesetzt werden, da diese Auflösung, für die Fälle, die ich untersucht habe, zur eindeutigen Diskriminierung meistens ausreicht. Aber auch alle anderen Werte sind möglich und durch den Anwender vorgebbar. Im Weiteren wird davon ausgegangen, dass die Segmente und Referenzen (ob zum Lernen oder Erkennen) immer normiert vorliegen.

2.6 Referenzen

2.6.1 Einleitung

Mit Hilfe der o.g. Merkmale lässt sich jedes Muster einfach codieren. Diese Technik funktioniert für Muster beliebiger Größe und kann natürlich auch für Teile des Gesamtbildes eingesetzt werden. Oben wurde der Schriftzug LEO komplett codiert. Genauso hätte man LEO in seine Buchstaben zerlegen und die Buchstaben einzeln codieren können.

Das bedeutet für Autokennzeichen, dass man ein digitales Foto eines Fahrzeuges mit Hilfe des Segmentierers „zerlegt“ und für jedes Segment die Merkmale errechnet und speichert bzw. lernt. Es besteht natürlich auch die Möglichkeit, das Autokennzeichen als Ganzes zu erfassen, zu codieren und zu speichern. Um dann ein Autokennzeichen später wieder zu erkennen, müsste jedes vorkommende Autokennzeichen einmal gelernt werden; eine recht aufwändige Prozedur. Da bei den Autokennzeichen aber letztlich nur die Großbuchstaben A bis Z, die Ziffern 0 bis 9 und die Umlaute Ä,Ö und Ü – also 37 Zeichen – vorkommen, liegt es auf der Hand, nur die zu erkennenden Buchstaben und Ziffern zu lernen. Dazu dürfte dann eine recht geringe Auswahl von Bildern genügen, um alle Buchstaben und Ziffern abzudecken. Für Au-

tokennzeichen (aber auch andere) gibt es aber eine wesentlich elegantere Möglichkeit.

2.6.2 Lernen aus Fonts

Bereits die „alten“ Autokennzeichen wurden mit Hilfe einer Standardschrift (**DIN 1451**) erzeugt. Die neuen Euro-Kennzeichen folgen der Schriftart **Cargo Two SF** [**KENNZ**],. die im Internet unter www.anke-art.de als Font für Windows **kostenfrei** erhältlich ist. Die Schriftart **DIN 1451** kann man im Internet gegen **eine Gebühr** z.B. bei www.myfonts.com/fonts/linotype/din-1451/mittelschrift/ oder **kostenfrei** bei www.mexxoft.de/pfschaller/ durch Klick auf Downloads herunterladen. Aber auch die Windows-Standardschrift **Arial** liefert schon gute Ergebnisse.

Mit Hilfe dieser Fonts bietet sich nun folgende Verfahrensweise an:
Man generiert einen String für jeden gewünschten Buchstaben/Umlaut und jede Ziffer. Anschl. generiert man je Zeichen ein Bitmap und „zeichnet“ jedes gewünschte Zeichen in der gewünschten Schriftgröße und der gewünschten Schriftart in das Bitmap. Anschließend wird dieses Bitmap codiert und gespeichert. Delphi stellt dafür entsprechende Eigenschaften und Methoden zur Verfügung:

(1) Font für die gewünschte Schriftart.
(2) Font.Size für die gewünschte Schriftgröße.
(3) Textout zum Zeichnen in ein Bitmap.
(4) SaveToFile zum Speichern des Bitmaps auf einem externen Speicher.

Man erhält so eine optimale Vorlage für die Referenzen, da die so generierten **monochromen** Muster keinerlei störenden Einflüssen wie unterschiedlichen Lichtverhältnisse, Schmutz, Verdrehungen, Unschärfen etc. ausgesetzt waren.

2.7 Speichern

2.7.1 Einleitung

Wie oben gezeigt, werden für jedes Segment die gewünschten Merkmale codiert. Dabei wird je Merkmal ein Vektor mit ganzen Zahlen generiert. Aus allen Merkmalsvektoren lässt sich ohne Problem ein „Gesamtvektor“ generieren. Dessen Länge entspricht der Summe der Längen aller Einzelvektoren.

Zur Speicherung dieses Gesamtvektors gibt es nun verschiedene Möglichkeiten. In dieser Arbeit wird von einer Speicherung in künstlichen neuronalen Netzen ausgegangen, wobei eine spezielle Form der neuronalen Netze, die so genannten spärlich codierten Assoziativmatrizen (SpaCAM) eingesetzt werden. Die nächsten Abschnitte sollen in diese Technik einführen.

2.7.2 Künstliche neuronale Netze

Ein neuronales Netz ist durch das Tupel (U, W, ex, NET, A, O) spezifiziert [**HEIT1994**, Kapitel 2.2]:

- $U = \{u_1,...,u_k\}$ ist die Menge von k Verarbeitungseinheiten (units), die als (neuronale) Einheiten bezeichnet werden.
- W ist die Netzwerkstruktur, dargestellt durch eine Konnektionsmatrix oder durch einen gerichteten Graphen mit bewerteten Kanten.
- ex ordnet jeder unit $u_e \in U$ eine Funktion $ex_e(t)$ zu, um die externe Eingabe in u_e zum Zeitpunkt t zu ermitteln.
- NET ist die Funktion, die jeder Einheit $u_j \in U$ eine Netzeingabefunktion (Propagierungsfunktion) $net_j(t)$ zur Berechnung der aktuellen Eingabe zuordnet. Zum Zeitpunkt t berechnet sich diese Eingabe anhand der Ausgabesignale der vorgeschalteten Einheiten $o_i(t)$ und den entsprechenden Verbindungsgewichten w_{ij}. Die gebräuchlichste Propagierungsfunktion ist die Summenfunktion $net_j(t) = \sum_i w_{ij} o_i(t) + \theta_j$, wobei θ_j den negativen Schwellenwert oder so genannten Bias der Einheit u_j darstellt. Dieser lässt sich als expliziter Ausdruck in der Notation vermeiden, in dem $w_{0j} = \theta_j$ und $o_0(t) = 1$ vereinbart wird. Bei einer weiteren Klasse von Propagierungsfunktionen wird keine additive, sondern eine multiplikative Verknüpfung der Aufgaben vorgenommen. Ein wichtiger Vertreter dieser Klasse ist die so genannte Sigma-Pi-Propagierungsfunktion $net_j(t) = \sum_{i=1}^{p} w_{ij} \prod_{q=1}^{q_i} o_{pq}(t)$ wobei p die Anzahl der multiplikativen Verbindungen (Konjunkte) ist und q_i die Anzahl der dem i-ten Konjunkt verknüpften Einheiten.

- A ordnet jeder Einheit $u_j \in U$ eine Aktivierungsfunktion F_j zur Berechnung eines neuen Aktivierungszustandes a_j zu. Aktivierungsfunktionen können abhängen von der aktuellen Aktivierung, sowie von externen und internen Netzeingaben:
  $a_j(t+1) = F_j(a_j(t), ex_j(t), net_j(t))$
- O ordnet jeder Einheit $u_j \in U$ eine Ausgabefunktion f_j zur Berechnung der aktuellen Ausgabe o_j in Abhängigkeit ihres aktuellen Zustands $a_j(t)$ zu; d.h. $o_j(t)=f_j(a_j(t))$

Für weitere Informationen über neuronale Netze siehe [**HEIT1994]** und auch [**RITT1991**].

2.7.3 SpaCAM

Die spärlich codierten Assoziativspeicher (Sparsely Coded Associative Memory; kurz SpaCAM) können als eine Sonderform der neuronalen Netze aufgefasst werden, die folgende Eigenschaften besitzen (zu dem gesamten Kapitel 2.7.3 vgl. [**HEIT1994**, Kapitel 2.3]):

- Die Netzstruktur weist zwei Schichten auf.
- Die Propagierungsfunktion der Netzaktivität erfolgt nur vorwärtsgerichtet.
- Die Lernregel ist Hebb-ähnlich.
- Es werden ausschließlich binäre Synapsen verwendet.
- Zur Abbildung des Musters auf die Eingangsneuronen werden ähnlichkeitserhaltende Codierungsfunktionen benutzt.
- Ein- und Ausgangscodierung sind spärlich, das heißt, dass der Anteil der aktivierten Neuronen gering ist.
- Die Neuronen der Eingangsschicht sind vollständig mit den Neuronen der Ausgangsschicht verbunden.

Zur Erläuterung der Arbeitsweise ein einfaches Beispiel:
Eine Anfrage von „HARD“ soll zur Ausgabe von „WORK“ führen. Implementiert man dies nicht wort- sondern zeichenweise, so müssen die Musterpaare
$\{("H","W"),("A","O"),("R","R"),("D","K")\}$ gelernt werden.

Konventionell könnte man dies dadurch tun, dass man die Zeichen untereinander in einer Tabelle speichert und zwar die Anfragezeichen in den ungeraden und die Ausgabezeichen in den geraden Zeilen. Dies ist ein sog. Listenspeicher.

H
W
A
O
R
R
D
K

Wird z.B. für jedes Zeichen der binäre Vektor entsprechend dem 8-bit ASCII-Code abgeleitet – dies ist die Merkmalsextraktion und Merkmalscodierung – so ergibt dies eine binäre 8x8 Matrix mit einem Speicherbedarf von 8 Byte. Wie man leicht erkennt, wächst hierbei der Speicher mit der Anzahl zu speichernder Musterpaare. Eine Anfrage würde das Suchmuster in den ungeraden Zeilen suchen und anschl. die Zeile darunter als Ergebnis der Anfrage ausgeben.

H		0	1	0	0	1	0	0	0
	W	0	1	0	1	0	1	1	1
A		0	1	0	0	0	0	0	1
	O	0	1	0	0	1	1	1	1
R		0	1	0	1	0	0	1	0
	R	0	1	0	1	0	0	1	0
D		0	1	0	0	0	1	0	0
	K	0	1	0	0	1	0	1	1

Nun soll eine SpaCAM erzeugt werden. Dazu wird zunächst eine leere (d.h. nur Nullen enthaltene) binäre 8x8 Matrix A erzeugt.

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Nun wird jedes Musterpaar (auch Frage-/Antwortpaar genannt) durch den ASCII-Code auf das binäre Eingabe-/Ausgabevektorpaar abgebildet und gelernt. („H“,“W“) wird damit auf ((0,1,0,0,1,0,0,0), (0,1,0,1,0,1,1,1)) abgebildet und gelernt, in dem die Ausgabevektoren sukzessive in die Spalten der Matrix A eingetragen werden, bei denen der zugehörige Eingabevektor eine 1 enthält – die Neuronen werden aktiviert bzw. es wird eine synaptische Verbindung hergestellt, wenn Eingabe- und Ausgabevektor beim bitweisen Vergleich eine gemeinsame 1 haben. Sollte das Matrixelement bereits 1 sein, so wird dieser Wert beibehalten. Umgekehrt hat/behält die Matrix genau dort eine Null, wo keines der gelernten Musterpaare eine Assoziation an dieser Stelle aufweist.

Sei $\vec{e}$ der binär codierte Eingabevektor und $\vec{a}$ der binär codierte Ausgabevektor, so berechnet sich die 8x8 Matrix A nach folgender Regel, wobei die Komponenten A_{ij}, e_i und a_j binär und die Operatoren boolesche sind:

$$A_{ij_{neu}} \Leftarrow A_{ij} \vee (e_i \wedge a_j) \quad i, j = 0,1,...,7$$

Das Lernen des ersten Musterpaares („H“,“W“) ergibt dann:

„H“	0	**1**	0	0	**1**	0	0	0	
	0	**0**	0	0	**0**	0	0	0	0
	0	**1**	0	0	**1**	0	0	0	1
	0	**0**	0	0	**0**	0	0	0	0
	0	**1**	0	0	**1**	0	0	0	1
	0	**0**	0	0	**0**	0	0	0	0
	0	**1**	0	0	**1**	0	0	0	1
	0	**1**	0	0	**1**	0	0	0	1
	0	**1**	0	0	**1**	0	0	0	1
									„W“

Nun wird das Musterpaar („A“,“O“) dazugelernt.

“A”	0	**1**	0	0	0	0	0	**1**		
“H”	0	**1**	0	0	1	0	0	**0**		
		0			0			**0**	0	0
		1			1			**1**	1	1
		0			0			**0**	0	0
		1			1			**0**	1	0
		1			0			**1**	0	1
		1			1			**1**	1	1
		1			1			**1**	1	1
		1			1			**1**	1	1
									“W”	“O”

Wie man sieht, zeigt die zweite Spalte Überlagerungen beider Musterpaare, da in der Matrixspalte bereits teilweise eine 1 eingetragen war.

Nach dem Lernen aller 4 Musterpaare ergibt sich die vollständige Assoziativmatrix:

“D”	0	1	0	0	0	1	0	0				
“R”	0	1	0	1	0	0	1	0				
“A”	0	1	0	0	0	0	0	1				
“H”	0	1	0	0	1	0	0	0				
	0	0	0	0	0	0	0	0	0	0	0	0
	0	1	0	1	1	1	1	1	1	1	1	1
	0	0	0	0	0	0	0	0	0	0	0	0
	0	1	0	1	1	0	1	0	1	0	1	0
	0	1	0	0	0	1	0	1	0	1	0	1
	0	1	0	0	1	0	0	1	1	1	0	0
	0	1	0	1	1	1	1	1	1	1	1	1
	0	1	0	0	1	1	0	1	1	1	0	1
									“W”	“O”	“R”	“K”

Nun soll untersucht werden, ob und wie eine Anfrage von “R” auf den gesuchten Ausgabewert von “R” führt. Dabei wird „R“ entsprechend dem ASCII-Code auf den Anfragevektor $\vec{e} = (0,1,0,1,0,0,1,0)$ codiert. Dann wird die Assoziativmatrix A mit dem (transponierten) Anfragevektor $\vec{e}^T$ multipliziert. Das Ergebnis ist der Schwellvektor $\vec{s}$.

A									$\vec{e}^T$		$\vec{s}$
0	0	0	0	0	0	0	0		0		0
0	1	0	1	1	1	1	1		1		3
0	0	0	0	0	0	0	0		0		0
0	1	0	1	1	0	1	0	->	1	->	3
0	1	0	0	0	1	0	1		0		1
0	1	0	0	1	0	0	1		0		1
0	1	0	1	1	1	1	1		1		3
0	1	0	0	1	1	0	1		0		1

Da die Matrix binär codiert ist, ist im mathematischen Sinn keine Matrixmultiplikation nötig, sondern es genügt das Aufsummieren weniger Merkmalsspalten.

Nun wird der Schwellvektor $\vec{s}$ mit dem Schwellwert σ abgeglichen. Alle Schwellvektoreinträge größer oder gleich σ werden auf 1 und alle anderen auf 0 abgebildet. Dies ist dann der Ausgangsvektor $\vec{o}$.

Wird nun σ auf die Summe der Einsen im Anfragevektor $\vec{e}$ gesetzt (in diesem Fall σ = 3), so ergibt sich der Ausgabevektor zu $\vec{o} = (0,1,0,1,0,0,1,0)$ und das entspricht dem „R“ im ASCII-Code. Alle anderen Anfragen für die Musterpaare laufen analog.

Bei der Listenspeicherung (s.o.) hängt der Speicherplatz der Matrix von der Länge **und** der Anzahl der Musterpaare ab, bei der Assoziativmatrix nur von der Länge der Musterpaare. Mit zunehmender Anzahl von Musterpaaren verschlechtert sich allerdings allmählich die Erkennungsfähigkeit, da sich die Matrix zunehmend mit Einsen füllt. Das Eingangsmuster wird zwar stets gefunden, aber auch mehr oder weniger viele nicht zusammen mit dem Eingangsmuster gelernte Ausgangsmuster. Das exakte Verhalten der SpaCAM hängt von der Anzahl der Muster und von der Dichte der Einsen im Eingangs- und Ausgangsvektor ab. In der Praxis hat sich gezeigt, dass sich in Bezug auf Speicherausnutzung, Zeitaufwand und Musterdiskriminierung spärliche Codierungen als günstig erweisen. Bei der spärlichen Codierung ist nur ein ge-

ringer Teil der Vektoreinträge (möglichst gleichverteilt) auf 1 gesetzt. Weiterhin sollten die Codierungen ähnlichkeitserhaltend sein, um eine fehlertolerante Erkennung zu ermöglichen. Die o.g. Merkmalscodierung mit dem ASCII-Code erfüllt diese Anforderungen nicht.

Nun soll ein weiterer Vorzug der SpaCAM gezeigt werden.
Angenommen, die Anfrage von „R“ wird fehlerhaft mit $\vec{e} = (0,1,0,0,0,1,0)$ codiert.

A									$\vec{e}^T$		$\vec{s}$
0	0	0	0	0	0	0	0		0		0
0	1	0	1	1	1	1	1		1		2
0	0	0	0	0	0	0	0		0		0
0	1	0	1	1	0	1	0	->	0	->	2
0	1	0	0	0	1	0	1		0		1
0	1	0	0	1	0	0	1		0		1
0	1	0	1	1	1	1	1		1		2
0	1	0	0	1	1	0	1		0		1

Wird nun wieder zunächst der Schwellwert σ auf die „richtige“ Anzahl Einsen im Anfragevektor gesetzt (also σ = 3) so wird der Ausgabevektor $\vec{o} = (0,0,0,0,0,0,0)$, da im Schwellvektor kein Wert ≥ 3 enthalten ist. Der Fehler im Anfragevektor wird also erkannt. Wird σ = 2 (der Anzahl Einsen im „falschen“ Anfragevektor) gesetzt, ergibt sich wiederum als Ausgangsvektor $\vec{o} = (0,1,0,1,0,0,1,0)$. Damit wird trotz eines Fehlers im Eingangsvektor der korrekte Ausgangsvektor gefunden.

Das gleiche gilt auch, wenn beim **Lernen** der Assoziativmatrix der Eingangsvektor für “R“ falsch codiert wurde. Der korrekte Ausgangsvektor wird in diesem Beispiel trotzdem gefunden.

Die SpaCAM kann somit Fehler beim Lernen als auch bei der Anfrage ausgleichen. Sie ist also im hohen Maße fehlertolerant. Auch nachträgliche Veränderungen an der

Matrix (z.B. durch „Umkippen“ von Bits, technische Defekte, teilweise Zerstörung o.ä.) kann die Matrix in einem gewissen Bereich ausgleichen.

Aus der Theorie der neuronalen Netze handelt es sich bei der SpaCAM um ein vorwärtsgetriebenes zweischichtiges Netzwerk, wobei jedes Eingangsneuron mit jedem Ausgangsneuron synaptisch verbunden ist. Die Werte der Synapsen werden in der Lernphase aus den zu lernenden Mustern ermittelt und in einer binären Konnektionsmatrix (Assoziativmatrix) gespeichert.

Allgemein wird die Assoziativmatrix folgendermaßen berechnet:
Seien N Musterpaare (P_k^e, P_k^a) gegeben. Mittels Merkmalsextraktion und –codierung werden sie auf N Paare binärer Vektoren (e_k, a_k) der Dimension n bzw. z abgebildet. Das äußere Produkt jedes Vektorpaares bildet eine binäre z x n Matrix $(m_{ij}^k) = \vec{a}_k^T \vec{e}_k$.
Die Assoziativmatrix A aller Musterpaare kann auf verschiedene Arten aus den einzelnen Matrizen abgeleitet werden:

(1) $A = (a_{ij})\ mit\ a_{ij} = \sum_{k=0}^{N-1} w^k m_{ij}^k$, wobei w^k reelle Gewichtsfaktoren sind

(2) $A = (a_{ij})\ mit\ a_{ij} = \sum_{k=0}^{N-1} m_{ij}^k$

(3) $A = (a_{ij})\ mit\ a_{ij} = \min(1, \sum_{k=0}^{N-1} m_{ij}^k)$

Die dritte Variante entspricht einer logischen *Oder*-Verknüpfung aller N Matrizen und ist aus mehreren Gründen den anderen Berechnungsvorschriften vorzuziehen:

- Hohe Speicherausnutzung: unabhängig von der Zahl der gelernten Musterpaare wird zur Darstellung der Synapsenwerte stets nur ein Bit benötigt, während bei den anderen genannten Verfahren ganzzahlige oder rationale Werte abgelegt werden müssen. Es lässt sich zeigen, dass diese Methode im Vergleich zu allen anderen Speicherverfahren den Speicherplatz am besten ausnutzt.
- Auch ein mehrfaches Lernen des selben Musterpaares verändert die Matrix nicht. Überprüfungen auf Mehrfachmuster sind damit überflüssig.

- Die beim Retrieval durchzuführenden Matrixmultiplikationen reduzieren sich auf ein effizient durchzuführendes Aufsummieren weniger Matrixspalten. Statt aufwändiger Multiplikationen rationaler Zahlen wird lediglich das als Maschinenbefehl verfügbare Inkrementieren benötigt. Dies führt damit zu einfachen, leicht portierbaren Programmen.
- Da die Rechenoperationen einfach sind, bietet es sich auch an, diese Technik hardwaremäßig zu realisieren.

In der Applikation gibt es einen Programmteil, mit dem o.g. Techniken mit unterschiedlichen Texten ausprobiert werden können. Anschl. sind Anfragen an die SpaCAM möglich. Sowohl das Lernen als auch die Anfrage werden visualisiert.

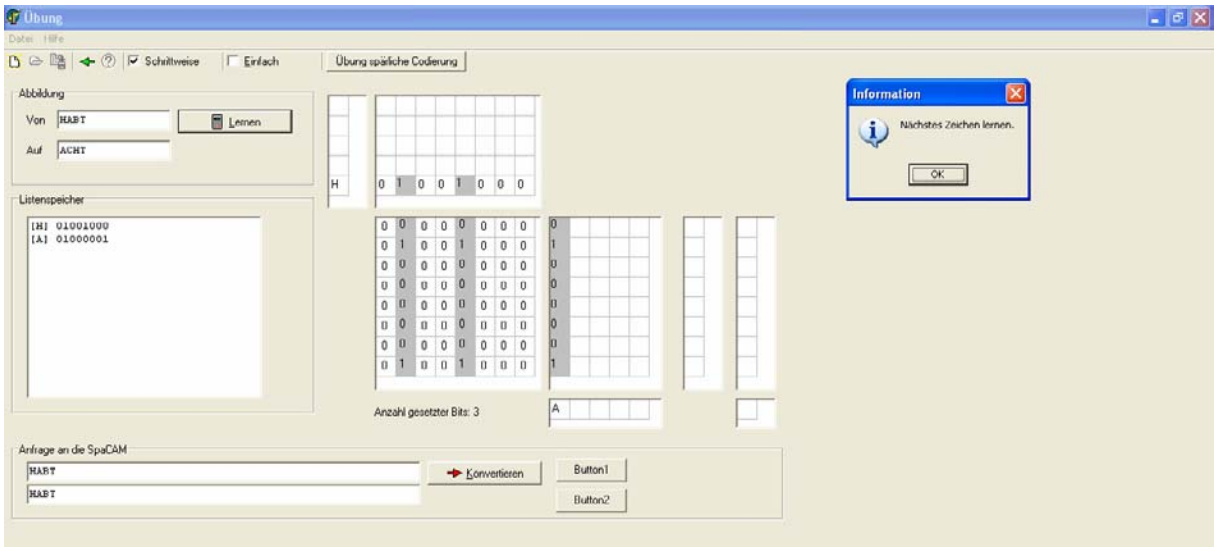


Abbildung 3 Übungsteil in der Applikation

Soweit zur Theorie der SpaCAM. Im Weiteren soll nun gezeigt werden, wie die Technik der SpaCAM für die Erkennung von Kfz-Kennzeichen nutzbringend eingesetzt werden kann.

2.8 Merkmalscodierung für die SpaCAM

2.8.1 Einleitung

Wie oben gezeigt, müssen die Merkmale binär und wenn möglich spärlich codiert werden. Bei den Beispielen mit den Buchstaben wurden die Merkmale mit Hilfe der ASCII-Codierung codiert.

Da die Merkmale SkyLine, Projection etc. rein numerische Vektoren sind, bietet sich in erster Näherung eine Codierung der Dezimalzahlen in Zahlen des dualen Systems an (z.B.: würde 10_{10} zu 1010_2). Im dualen System sind die Nullen und Einsen aber sicherlich nicht spärlich und schon gar nicht gleichverteilt.

Die spärliche Codierung wird daher so umgesetzt:

Definition:

Sei $n > 0$ und V ein Vektor mit m nicht negativen dezimalen Werten $V = (v_0, v_1, ..., v_{m-1})$ mit $0 \le v_i \le n \quad 0 \le i \le m-1$.

Weiterhin sei $V'_i = (v'_{i0}, v'_{i1}, ..., v'_{in}) \quad mit \quad v'_{i,j} = \begin{cases} 1, v_i = j \\ 0, v_i \neq j \end{cases} \quad 0 \le i \le m-1 \quad 0 \le j \le n$

Dann ist $V' = (V'_0, V'_1, ..., V'_{m-1})$ der spärlich codierte Vektor von V.

Ein Beispiel:

$n = 4, m = 3, V = (1,0,4)$

$v'_0 = (0,1,0,0,0)$

$v'_1 = (1,0,0,0,0)$

$v'_2 = (0,0,0,0,1)$

$V' = ((0,1,0,0,0,0),(1,0,0,0,0),(0,0,0,0,1))$

Für jede Komponente von V wird also ein Binärvektor der Länge n+1 gebildet. In diesem Binärvektor sind alle Bits 0 bis auf jenes, das dem Wert der entsprechenden Komponente von V entspricht.

Wie man sieht, ist V´ in jedem Fall binär. Die Länge des Vektors V´ ist m(n+1) und die Anzahl Einsen entspricht m. Folglich ist die relative Besetzung $\frac{m}{m(n+1)} = (n+1)^{-1}$.

Dies ist sicherlich spärlich; d.h. alle Vektoren V´ haben bei konstantem m und n die selbe Anzahl Einsen.

Besonderheit:

Wenn n=1 ist, gilt $v_i \in \{0,1\}$ - V ist also bereits ein **Binärvektor**. Allerdings muss V nicht spärlich codiert sein, da dort alle v_i=1 sein können.

Über o.g. Rechenregeln wird aber V ein spärlich codierter Vektor V´ (mit doppelt so vielen Komponenten wie V) durch

$$V'_{2i} = \begin{cases} 0, v_i = 1 \\ 1, v_i = 0 \end{cases} \quad i = 0,1,...,m-1$$

$$V'_{2i+1} = \begin{cases} 0, v_i = 0 \\ 1, v_i = 1 \end{cases} \quad i = 0,1,...,m-1$$

2.8.2 Vereinfachung

Liegen die Merkmalsvektoren spärlich codiert vor und ist zum **Lernen** der SpaCAM der Eingangs- und Ausgangsvektor gleich (bei den Autokennzeichen ist das sicherlich der Fall, da ein erkanntes „K“ auch als „K“ ausgegeben werden soll), so bietet es sich an, den Aufbau der SpaCAM zu vereinfachen [**BECK1997** Seite 26 ff].

Es gibt z binäre Eingangsvektoren der Länge n $\vec{e}^{\,i} = (e_0^i, e_1^i, ..., e_{n-1}^i), \quad i = 0,1,...,z-1$. Zu jedem Eingangsvektor $\vec{e}^{\,i}$ wird ein Antwortvektor $\vec{a}^{\,i}$ der Länge z definiert, der folgende Struktur hat:

$$\vec{a}^{\,i} = (a_0^i, a_1^i, ..., a_{z-1}^i), \quad a_v^i \begin{cases} 0, v \neq i \\ 1, v = i \end{cases} \quad 0 \le v \le z-1$$

Der Antwortvektor hat damit lediglich eine Eins und zwar an der Stelle, zu der der zugehörige Eingangsvektor gehört. Die Antwortvektoren verweisen somit auf die zugehörige Frage.

Die Frage-/Antwortpaare haben demnach folgende Form:

$$(\vec{e}^{\,0}, \vec{a}^{\,0}) = ((e_0^0, e_1^0, ..., e_{n-1}^0), (1,0,....,0))$$
$$(\vec{e}^{\,1}, \vec{a}^{\,1}) = ((e_0^1, e_1^1, ..., e_{n-1}^1), (0,1,...,0))$$
$$\vdots$$
$$(\vec{e}^{\,z-1}, \vec{a}^{\,z-1}) = ((e_0^{z-1}, e_1^{z-1}, ..., e_{n-1}^{z-1}), (0,0,.....,1))$$

Werden diese Fragen-Antwort-Paare gem. o.g. Rechenregel gelernt, so wird, da der i. Antwortvektor nur eine Eins im i. Element hat, die i. Frage in der i. Zeile der Matrix abgelegt.

Diese Matrix hat dann folgenden Aufbau:

$$A = \begin{pmatrix} e_0^0 & e_1^0 & \dots & e_{n-1}^0 \\ e_0^1 & e_1^1 & \dots & e_{n-1}^1 \\ \vdots & \vdots & \ddots & \vdots \\ e_0^{z-1} & e_1^{z-1} & \dots & e_{n-1}^{z-1} \end{pmatrix} = \begin{pmatrix} \vec{e}^{\,0} \\ \vec{e}^{\,1} \\ \vdots \\ \vec{e}^{\,z-1} \end{pmatrix}$$

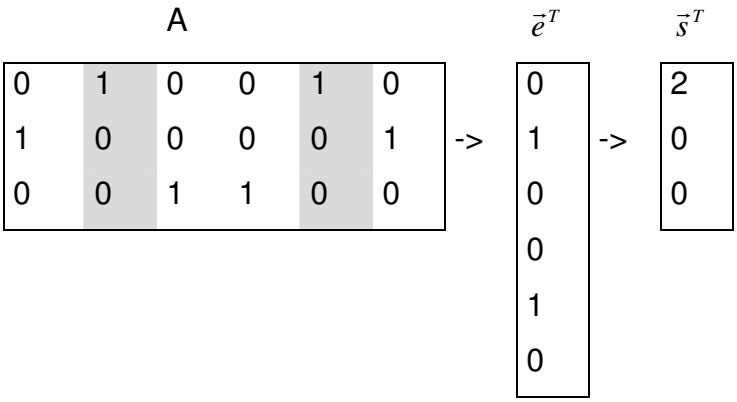
Wie man sieht, sind alle Eingangsvektoren einfach untereinander gespeichert. Dies kann ohne die o.g. Matrixoperationen durchgeführt werden. Zusätzlich wird der Speicherplatz deutlich reduziert. Die Anzahl der Spalten ist geblieben und zwar weiterhin so wie die Länge der Merkmalsvektoren. Die Anzahl der Zeilen reduziert sich aber auf die Anzahl gelernter Muster.

Ein Beispiel:

Es sollen die 3 spärlichen Vektoren $\vec{e}^{\,0} = (0,1,0,0,1,0)$, $\vec{e}^{\,1} = (1,0,0,0,0,1)$ und $\vec{e}^{\,2} = (0,0,1,1,0,0)$ gelernt werden. Die „zugehörigen“ Ausgabevektoren seien $\vec{a}^{\,0} = (1,0,0)$, $\vec{a}^{\,1} = (0,1,0)$ und $\vec{a}^{\,2} = (0,0,1)$. Dann stellt sich die SpaCAM so dar:

$\vec{e}^{\,2}$	0	0	1	1	0	0				
$\vec{e}^{\,1}$	1	0	0	0	0	1				
$\vec{e}^{\,0}$	0	1	0	0	1	0				
	0	1	0	0	1	0		1	0	0
	1	0	0	0	0	1		0	1	0
	0	0	1	1	0	0		0	0	1
								$\vec{a}^{\,0}$	$\vec{a}^{\,1}$	$\vec{a}^{\,2}$

Soll nun eine Anfrage mit dem Vektor $\vec{e}^T = (0,1,0,0,1,0)^T$ durchgeführt werden, so wird, wie bisher, die SpaCAM mit dem Anfragevektor multipliziert, um den Schwellvektor $\vec{s}$ zu erhalten:



Da die Zeilen der Assoziativmatrix den gelernten Mustern entsprechen, entspricht das Schwellvektorelement der Summe der übereinstimmenden Bits mit jedem der gelernten Muster. Die Position im Schwellvektor entspricht der Zeile in der Assoziativmatrix und ist damit ein Verweis auf die zugehörige Frage bzw. das zugehörige gelernte Eingangsmuster (das war durch die Konstruktion der Antwortvektoren so gewollt).

Nun kann man, wie bisher, den Schwellwert σ wieder auf die Anzahl Einsen im Eingangsvektor setzen (hier σ=2). Da lediglich s_0=σ ist, ist somit die „0. Zeile“ der Assoziativmatrix das Muster mit der „besten“ Übereinstimmung. Sortiert man den Schwellvektor absteigend nach seinen Komponenten und merkt sich dabei die ursprüngliche Position, so bekommt über den Schwellvektor eine abgestufte Übereinstimmung **aller** gelernten Muster mit dem Anfragevektor. Da die Anzahl Einsen in der Matrix je Zeile konstant ist, kann man über den Schwellvektor die relative Übereinstimmung des Anfragevektors mit **allen** gelernten Mustern ermitteln. Das bedeutet aber auch, dass man mit **einer** Matrixmultiplikation – die wiederum nur ein Aufsummieren einiger weniger Matrixspalten ist – die Übereinstimmung (absolut und relativ) des Eingangsvektors mit **allen** gelernten Mustern bekommt.

Nun soll die Umsetzung dieser Technik auf die o.g. Merkmale untersucht werden.

2.8.3 SkyLine, Projection und GroundLine

Die Vektoren SkyLineX, X-Projection und GroundLine haben so viele Komponenten, wie das Bitmap breit ist (B.w). Die Werte der Vektorkomponenten liegen alle im Bereich zwischen 0 und der Höhe des Bitmaps B.h einschließlich.
Bei der SkyLineY und der Y-Projection ist es „umgekehrt“. Die Länge entspricht B.h und die Vektorkomponenten liegen im Bereich zwischen 0 und B.w einschließlich. Diese Vektoren werden, wie oben beschrieben, spärlich codiert.

2.8.4 Gitter

Die Länge der Vektoren und die Werte der Vektorkomponenten hängen von der Anzahl Zeilen und Spalten des übergelegten Gitters ab. Da diese Parameter aber bekannt sind, können die Vektoren, wie oben geschildert, spärlich codiert werden.

2.8.5 Pixelvergleich

Der Vektor des Pixelvergleichs ist bereits ein Binärvektor und kann wie oben beschrieben, spärlich codiert werden.

2.8.6 Schwerpunkt

Der Schwerpunkt C_B des Musters ist unabhängig von seiner Größe und hat einen Wert von $0 \le C_B \le 8$. Somit wird z.B. der Schwerpunkt C_B=1 in einen Vektor $C'_B = (0,1,0,0,0,0,0,0,0)$ spärlich codiert.

2.8.7 Gesamtvektor

Alle einzelnen spärlich codierten Merkmalsvektoren können zu einem spärlich codierten Gesamtvektor des Bitmaps V_B zusammengefasst werden. Die Länge dieses Vektors entspricht der Summe der Länge der Einzelvektoren.

Da die Segmente und die Referenzen alle normiert vorliegen, haben die Gesamtvektoren aller Segmente und Referenzen auch dieselbe Länge und dieselbe spärliche Anzahl und Verteilung der Einsen.

2.8.8 Erzeugung der SpaCAM

Da der Anfrage- und Ausgabevektor gleich ist, kann die SpaCAM, wie oben geschildert, gelernt werden, in dem alle Referenzen einfach „untereinander“ gespeichert werden. Die Breite der SpaCAM entspricht dann der Länge des Gesamtvektors und die Höhe der Anzahl gelernter Referenzen.

2.8.9 Gruppierung

Liegen verschiedene Referenzen für ein zu erkennendes Muster vor (z.B. verschiedene „A“ unterschiedlicher optischer Qualität, wenn nicht mit Referenz-Fonts gearbeitet werden kann) so bietet es sich **zusätzlich** an, diese Referenzen zu einer Klasse zusammenzufassen und **je Klasse** aus den spärlich codierten Referenzen eine „Mittelwertreferenz“ zu bilden. Diese Mittelwertreferenzen lassen sich dann wieder mit den oben beschriebenen Techniken zu einer (ggf. eigenen) SpaCAM zusammenfassen. Eine Anfrage an diese SpaCAM gibt dann als **Gruppenvektor** eine Abstufung der Übereinstimmung in den Klassen zurück. Technisch können der Gruppen- und der Schwellvektor gleichzeitig berechnet werden. Somit lassen sich mit **einer** Anfrage Ergebnisse auf Gruppen- als auch auf Musterebene ermitteln (vgl. auch [**BECK1997**]).

2.9 Diskriminierung der Merkmale

2.9.1 Einleitung

Mit Hilfe der beschriebenen Merkmale soll nun zunächst untersucht werden, wie sich die einzelnen Buchstaben bzw. Ziffern unterscheiden bzw. ähneln. Dazu werden alle Buchstaben und Ziffern in der Schriftart der Autokennzeichen (**Cargo Two SF)**, normiert z.B. auf 32x32 Pixel, in den einzelnen Merkmalen codiert, die Merkmalsvektoren paarweise verglichen und daraus die relative Übereinstimmung ermittelt. Diese wird berechnet als die Anzahl gleicher Komponenten in den beiden Merkmalsvektoren geteilt durch die Gesamtanzahl der Komponenten.

Beispiel:

Für Zeichen Z_1 sei der dezimale Merkmalsvektor $e_1^T = (10,\underline{5},9,\underline{9})$ und für Zeichen Z_2 $e_2^T = (8,\underline{5},10,\underline{9})$. Jeder Merkmalsvektor hat 4 Komponenten. Bei zwei übereinstimmen-

den Komponenten (unterstrichen) ergibt sich eine relative Übereinstimmung von $\frac{2}{4} = 50\%$ [4].

Bei 26 Buchstaben, 3 Umlauten und 10 Ziffern müssen dazu $\frac{39 \cdot 38}{2} = 741$ Paare **je Merkmal bzw. Merkmalskombination** verglichen werden. Das Ergebnis eines jeden Vergleichs lässt sich als Verteilung graphisch oder tabellarisch darstellen. Dabei werden die Übereinstimmungen in Klassen zu je 5% zusammengefasst; also 0% bis < 5%, 5% bis < 10% usw. Aus Gründen der Übersichtlichkeit werden lediglich die Paare mit einer **Übereinstimmung** ab 85% tabelliert. In der Applikation können jederzeit diese Analysen über /**Bearbeiten**/**Merkmale analysieren** durchgeführt werden.

[4] Analog gilt dies natürlich auch für die spärlich codierten Vektoren.

2.9.2 SkyLineX

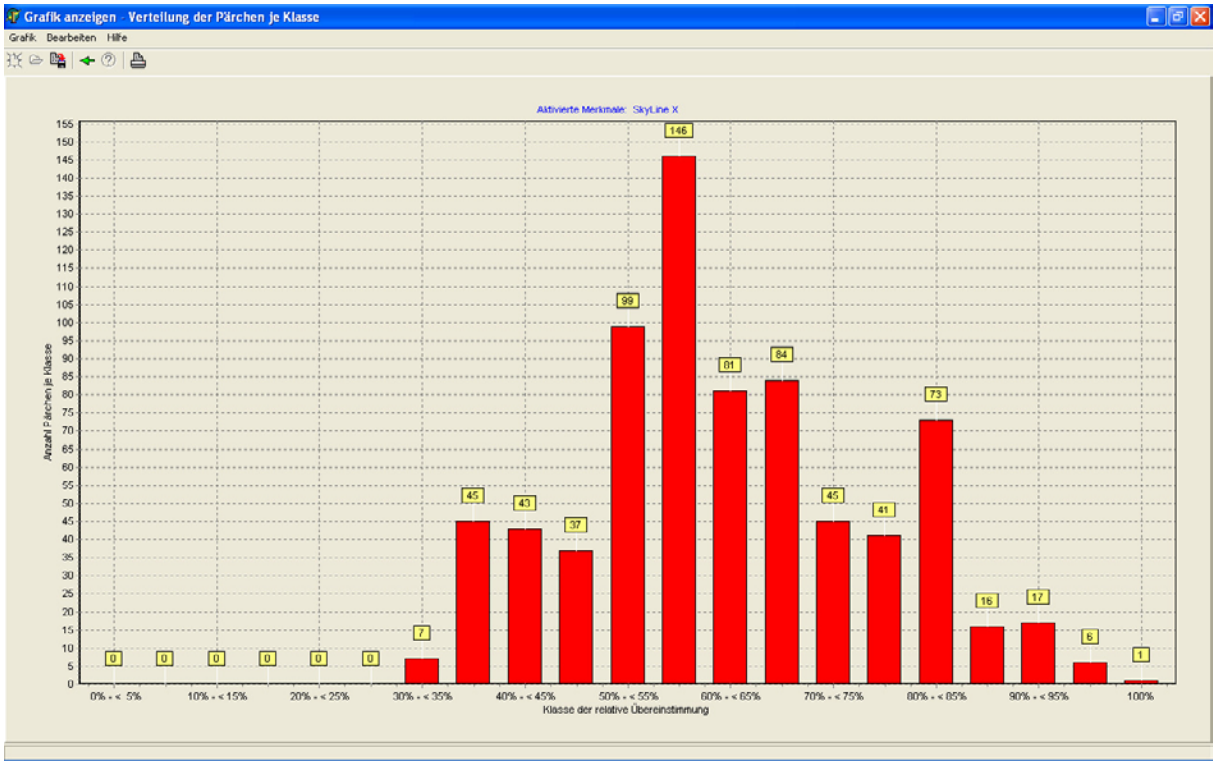


Abbildung 4 Verteilung der relativen Übereinstimmung der SkyLineX

Bei dem Merkmal SkyLineX sind einige Zeichenpaare gleich oder sehr ähnlich. Welche Zeichen nicht bzw. schlecht unterschieden werden können, kann der folgenden Tabelle entnommen werden.

Klasse	Anzahl	Paare
85% - < 90%	16	02 2Q 2R 3E 5D 5ü 7D 7ü CG DT DZ EF FQ FT Zü Äö
90% - < 95%	17	0Q 35 37 3T 3Z 5E 5R 7E 7R DR ET EZ FG GQ RT RZ Tü
95% - <100%	6	5T 5Z 7T 7Z BP TZ
100%	1	57

Tabelle 1 Ähnliche Zeichen beim Merkmal SkyLineX

2.9.3 SkyLineY

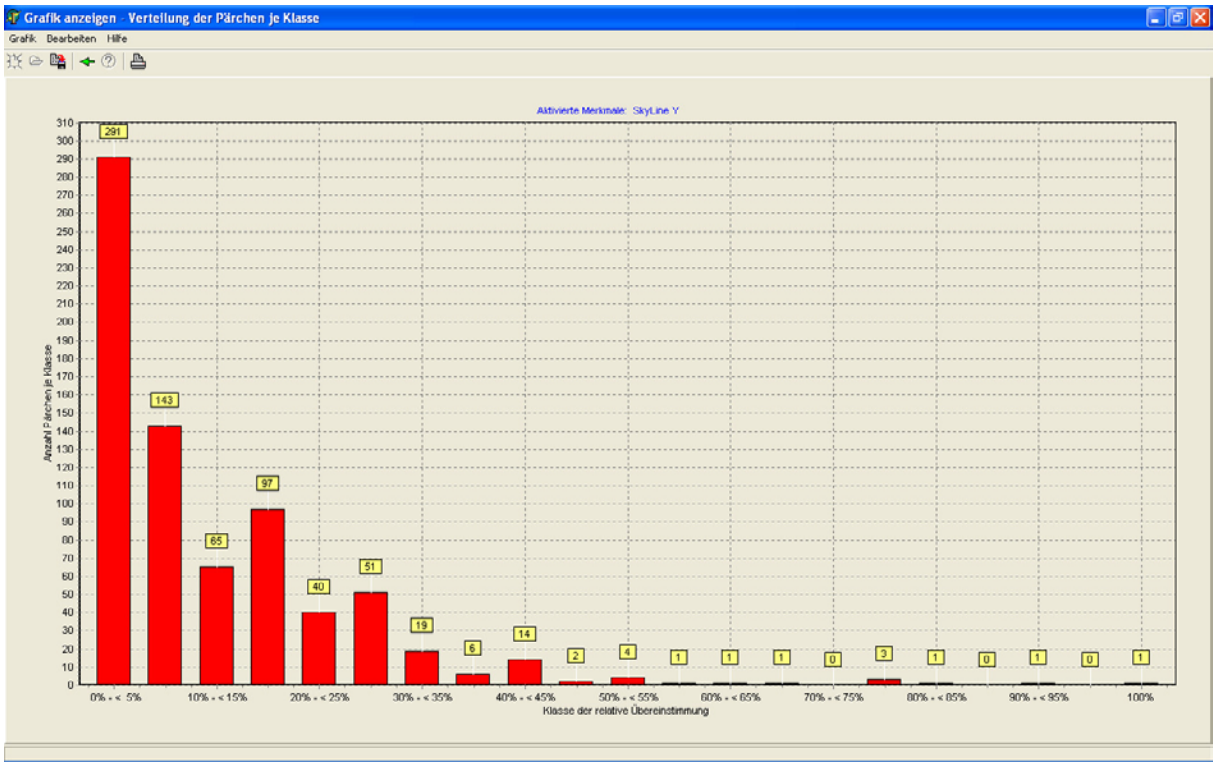


Abbildung 5 Verteilung der relativen Übereinstimmung der SkyLineY

Die SkyLineY diskriminiert die Zeichen erheblich besser. Für fast die Hälfte der Paare beträgt die relative Übereinstimmung nur 5% oder weniger. Lediglich J und U bzw. H und N können mit Hilfe dieses Merkmals kaum oder gar nicht unterschieden werden.

Klasse	An-zahl	Paare
85% - < 90%	0	
90% - < 95%	1	JU
95% - <100%	0	
100%	1	HN

Tabelle 2 Ähnliche Zeichen beim Merkmal SkyLineY

2.9.4 X-Projection

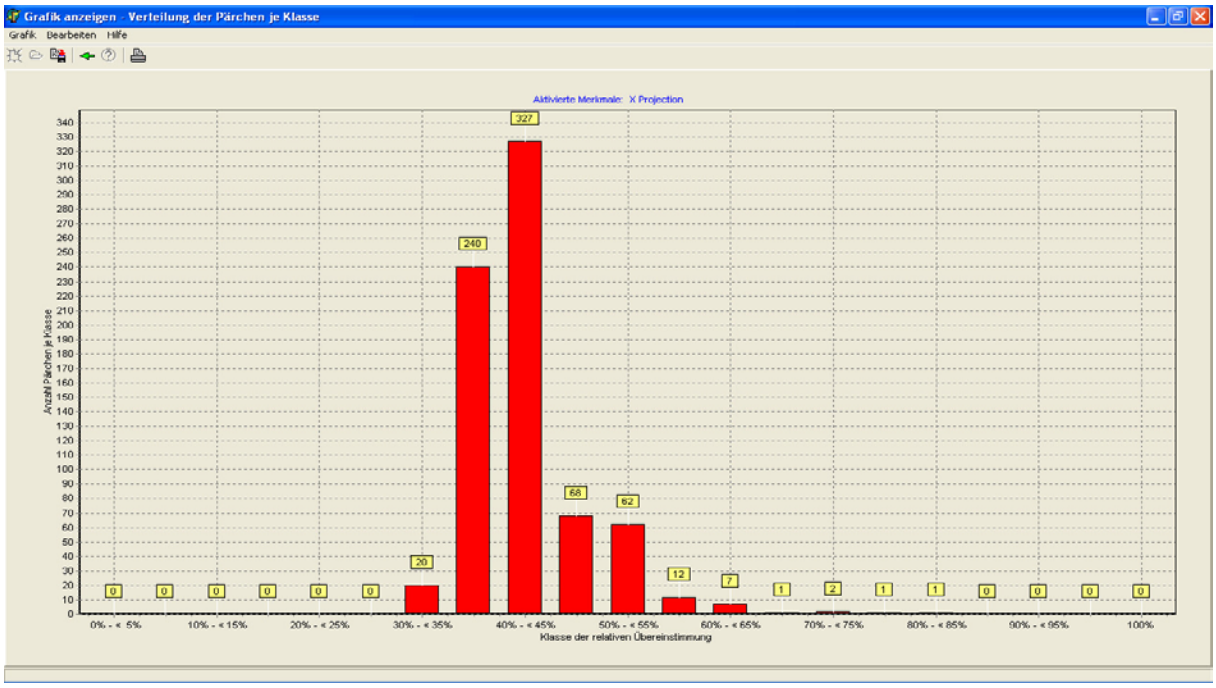


Abbildung 6 Verteilung der relativen Übereinstimmung der X-Projection

Auch hier ergibt sich eine gute Diskriminierung. Alle Paare haben eine Übereinstimmung unter 85%.

2.9.5 Y-Projection

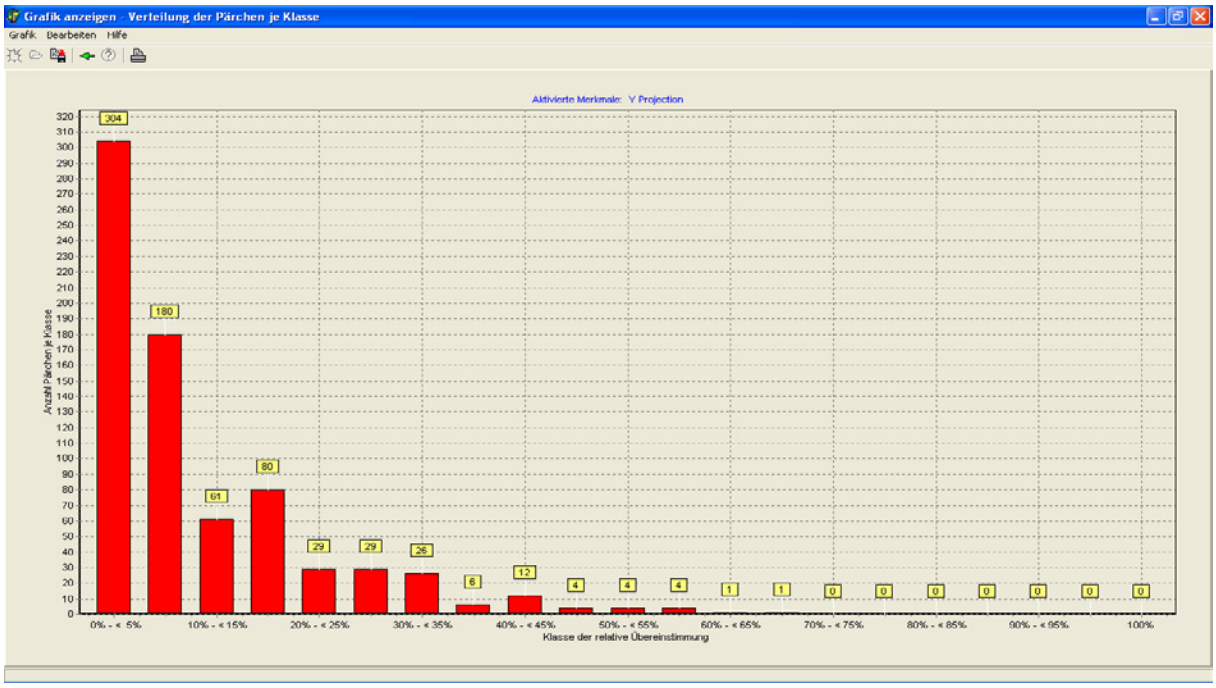


Abbildung 7 Verteilung der relativen Übereinstimmung der Y-Projection

Auch die Y-Projection lässt die Zeichen sehr gut unterscheiden; Zeichen mit gleicher oder stark ähnlicher Merkmalsausprägung gibt es nicht.

2.9.6 GroundLine

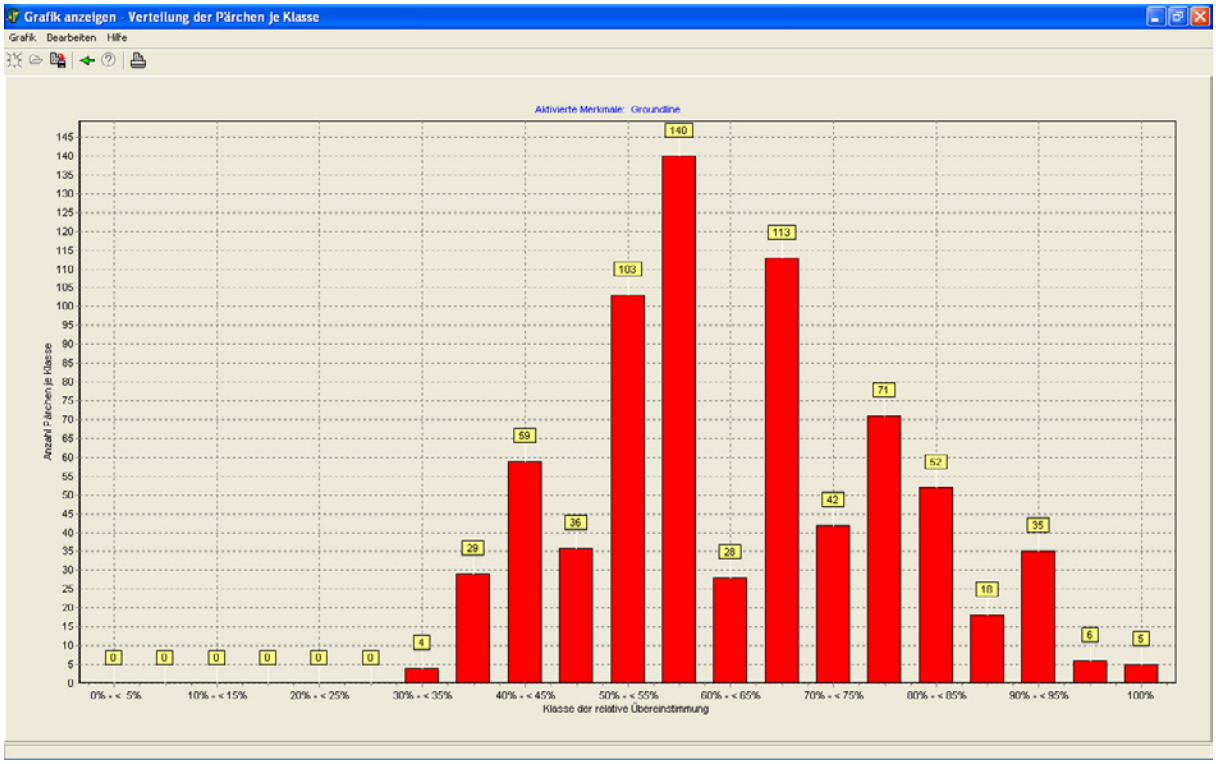


Abbildung 8 Verteilung der relativen Übereinstimmung der GroundLine

Die GroundLine eignet sich weniger gut zur Diskriminierung, da hier sehr viele Paare als ähnlich, 5 sogar als gleich erkannt werden.

Klasse	An-zahl	Paare
85% - < 90%	18	0B 0D 0E 0L 1U 2U BG BI DG DI EI Eü GJ IJ IL Jü Lü UZ
90% - < 95%	35	01 02 0I 0J 0Z 1B 1D 1G 1I 1J 1ü 2B 2D 2G 2I 2J 2ü 68 BE BJ BL BZ DE DJ DL DZ EG EJ GL GZ IZ JL JZ Sö Zü
95% - <100%	6	1E 1L 2E 2L EZ LZ
100%	5	12 1Z 2Z BD EL

Tabelle 3 Ähnliche Zeichen beim Merkmal GroundLine

2.9.7 Stabilität

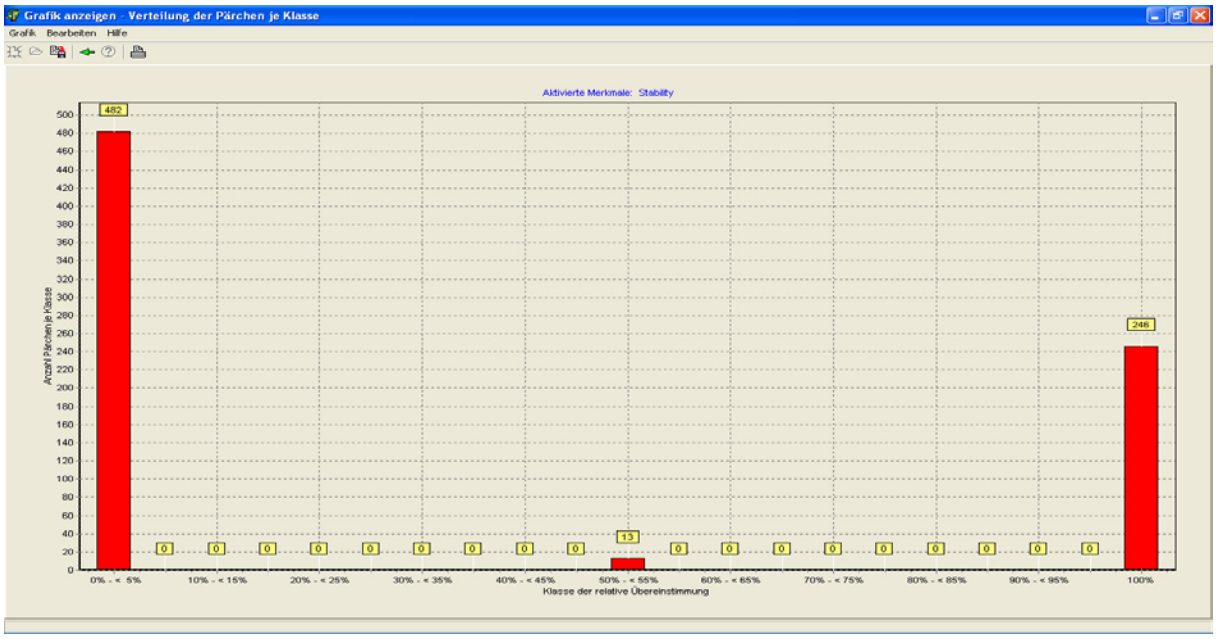


Abbildung 9 Verteilung der relativen Übereinstimmung der Stabilität

Die Stabilität ist für eine Diskriminierung praktisch ungeeignet, da ein Drittel der Zeichenpaare die selbe Stabilität haben.

Klasse	Anzahl	Paare
100%	246	04 05 06 07 08 12 13 19 1J 10 1P 1Q 1R 1T 1U 1Z 1ü 23 29 2J 20 2P 2Q 2R 2T 2U 2Z 2ü 39 3J 30 3P 3Q 3R 3T 3U 3Z 3ü 45 46 47 48 56 57 58 67 68 78 9J 90 9P 9Q 9R 9T 9U 9Z 9ü AB AC AD AE AF AG AH AK AL AN AS AV AW AX AY AÄ AÖ BC BD BE BF BG BH BK BL BN BS BV BW BX BY BÄ BÖ CD CE CF CG CH CK CL CN CS CV CW CX CY CÄ CÖ DE DF DG DH DK DL DN DS DV DW DX DY DÄ DÖ EF EG EH EK EL EN ES EV EW EX EY EÄ EÖ FG FH FK FL FN FS FV FW FX FY FÄ FÖ GH GK GL GN GS GV GW GX GY GÄ GÖ HK HL HN HS HV HW HX HY HÄ HÖ JO JP JQ JR JT JU JZ Jü KL KN KS KV KW KX KY KÄ KÖ LN LS LV LW LX LY LÄ LÖ NS NV NW NX NY NÄ NÖ OP OQ OR OT OU OZ Oü PQ PR PT PU PZ Pü QR QT QU QZ Qü RT RU RZ Rü SV SW SX SY SÄ SÖ TU TZ Tü UZ Uü VW VX VY VÄ VÖ WX WY WÄ WÖ XY XÄ XÖ YÄ YÖ Zü ÄÖ

Tabelle 4 Ähnliche Zeichen beim Merkmal Stabilität

2.9.8 Gitter

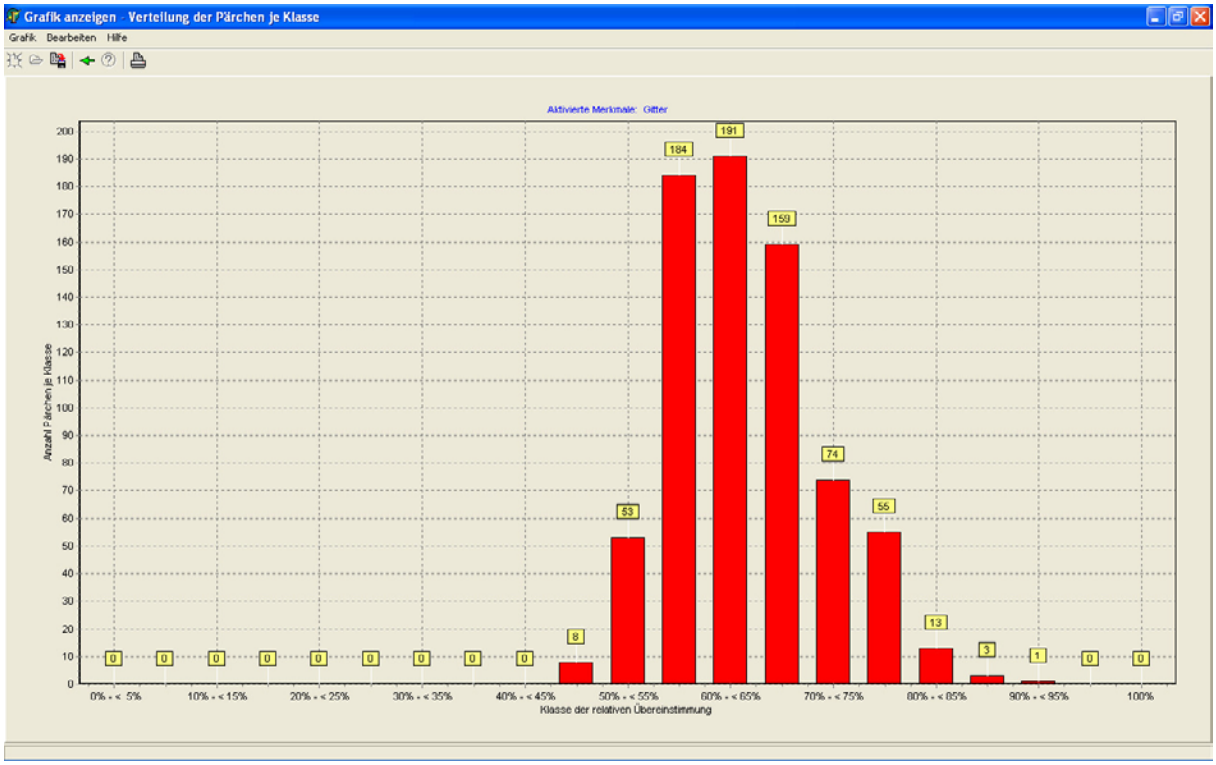


Abbildung 10 Verteilung der relativen Übereinstimmung des Gitters

Das Merkmal Gitter (hier mit 8 Spalten und 8 Zeilen; d.h. Felder zu 16 Pixel) diskriminiert die Zeichen wieder besser.

Klasse	Anzahl	Paare
85% - < 90%	3	OU HM HN
90% - < 95%	1	Uü
95% - <100%	0	
100%	0	

Tabelle 5 Ähnliche Zeichen beim Merkmal Gitter

2.9.9 Pixelvergleich

Der Pixelvergleich ist ein Gitter, bei dem die Anzahl der Spalten der Breite des Musters und die Anzahl der Zeilen der Höhe des Musters entspricht. Jedes Feld enthält ein Pixel.

Anmerkung:

Sowohl beim Gitter als auch beim Pixelvergleich werden nicht nur die gesetzten Pixel verglichen. Ein Merkmalswert ist auch dann gleich, wenn die Anzahl **nicht** gesetzter Pixel gleich ist.

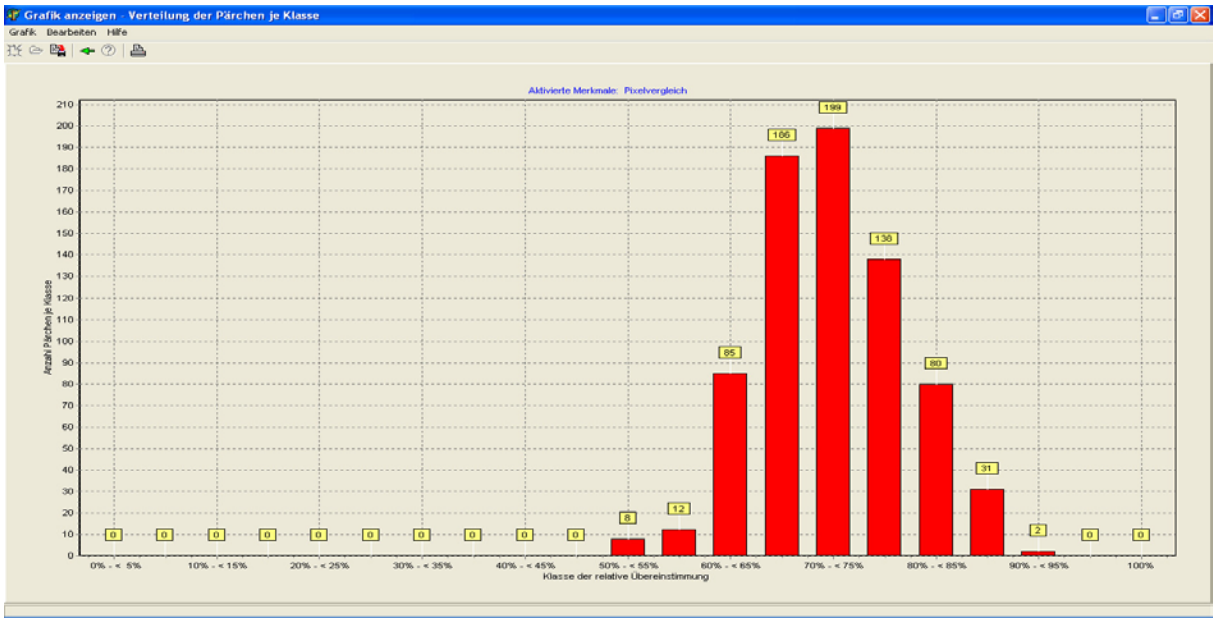


Abbildung 11 Verteilung der relativen Übereinstimmung des Pixelvergleichs

Beim Pixelvergleich ist die Diskriminierung etwas schlechter als bei o.g. Gitter.

Klasse	Anzahl	Paare
85% - < 90%	31	05 0B 0C 0G 0O 0Q 0U 0ü 2Z 68 7Z 8S AÄ BC BD BE BG BU CG CL CQ GU HM HU KR MN OQ QU Qü VY öü
90% - < 95%	2	HN Uü
95% - <100%	0	
100%	0	

Tabelle 6 Ähnliche Zeichen beim Merkmal Pixelvergleich

2.9.10 Schwerpunkt

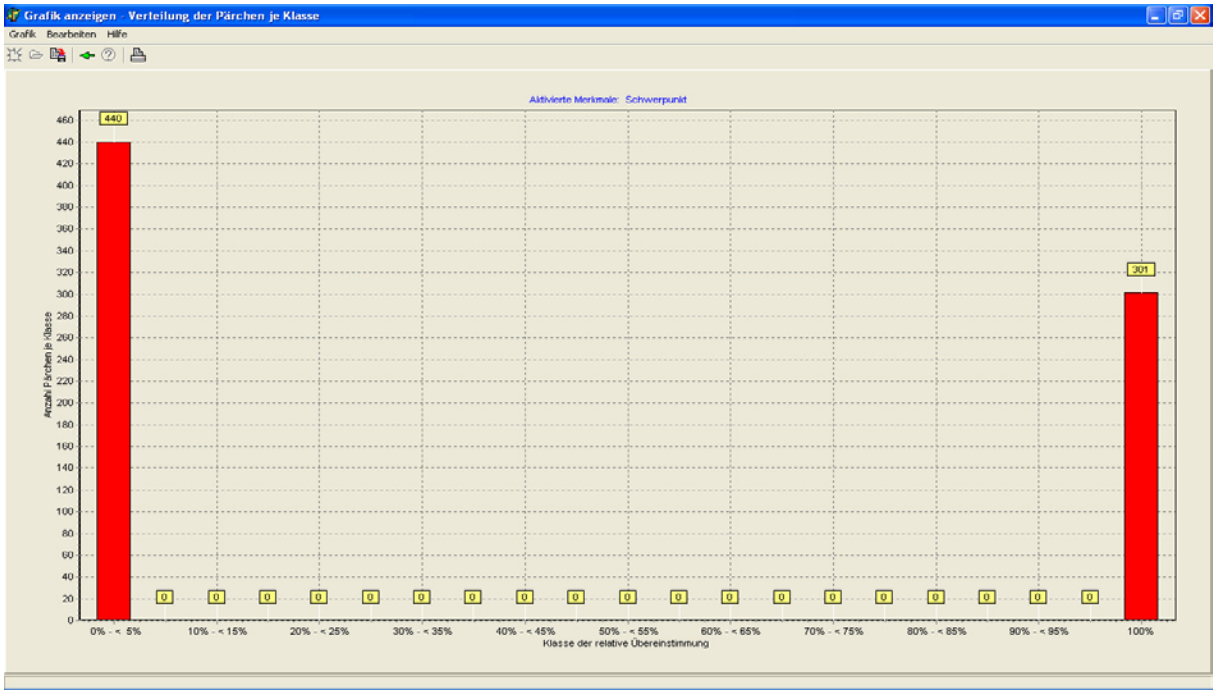


Abbildung 12 Verteilung der relativen Übereinstimmung des Schwerpunkts

Der Schwerpunkt ist für die Diskriminierung von Zeichen kein geeignetes Merkmal. Das ist auch wenig verwunderlich, da dieses Merkmal lediglich 9 Ausprägungen hat und von der Größe bzw. der Auflösung des Zeichens völlig unabhängig ist. Fast die Hälfte der Zeichenpaare stimmen zu 100% überein.

Klasse	Anzahl	Paare
100%	301	04 06 0A 0G 0I 0L 0U 0W 0Ä 1J 1ü 25 27 28 29 2B 2C 2D 2E 2F 2H 2K 2M 2O 2P 2R 2S 2T 2V 2X 2Y 2Z 2ö 46 4A 4G 4I 4L 4U 4W 4Ä 57 58 59 5B 5C 5D 5E 5F 5H 5K 5M 5O 5P 5R 5S 5T 5V 5X 5Y 5Z 5ö 6A 6G 6I 6L 6U 6W 6Ä 78 79 7B 7C 7D 7E 7F 7H 7K 7M 7O 7P 7R 7S 7T 7V 7X 7Y 7Z 7ö 89 8B 8C 8D 8E 8F 8H 8K 8M 8O 8P 8R 8S 8T 8V 8X 8Y 8Z 8ö 9B 9C 9D 9E 9F 9H 9K 9M 9O 9P 9R 9S 9T 9V 9X 9Y 9Z 9ö AG AI AL AU AW AÄ BC BD BE BF BH BK BM BO BP BR BS BT BV BX BY BZ Bö CD CE CF CH CK CM CO CP CR CS CT CV CX CY CZ Cö DE DF DH DK DM DO DP DR DS DT DV DX DY DZ Dö EF EH EK EM EO EP ER ES ET EV EX EY EZ Eö FH FK FM FO FP FR FS FT FV FX FY FZ Fö GI GL GU GW GÄ HK HM HO HP HR HS HT HV HX HY HZ Hö IL IU IW IÄ Jü KM KO KP KR KS KT KV KX KY KZ Kö LU LW LÄ MO MP MR MS MT MV MX MY MZ Mö OP OR OS OT OV OX OY OZ Oö PR PS PT PV PX PY PZ Pö RS RT RV RX RY RZ Rö ST SV SX SY SZ Sö TV TX TY TZ Tö UW UÄ VX VY VZ Vö WÄ XY XZ Xö YZ Yö Zö

Tabelle 7 Ähnliche Zeichen beim Merkmal Schwerpunkt

2.9.11 SkyLineY und Y-Projection kombiniert

Die SkyLineY und die Y-Projektion diskriminieren in meinen Versuchen die Zeichen am besten, deshalb habe ich entschieden, diese beiden Merkmale zu kombinieren.

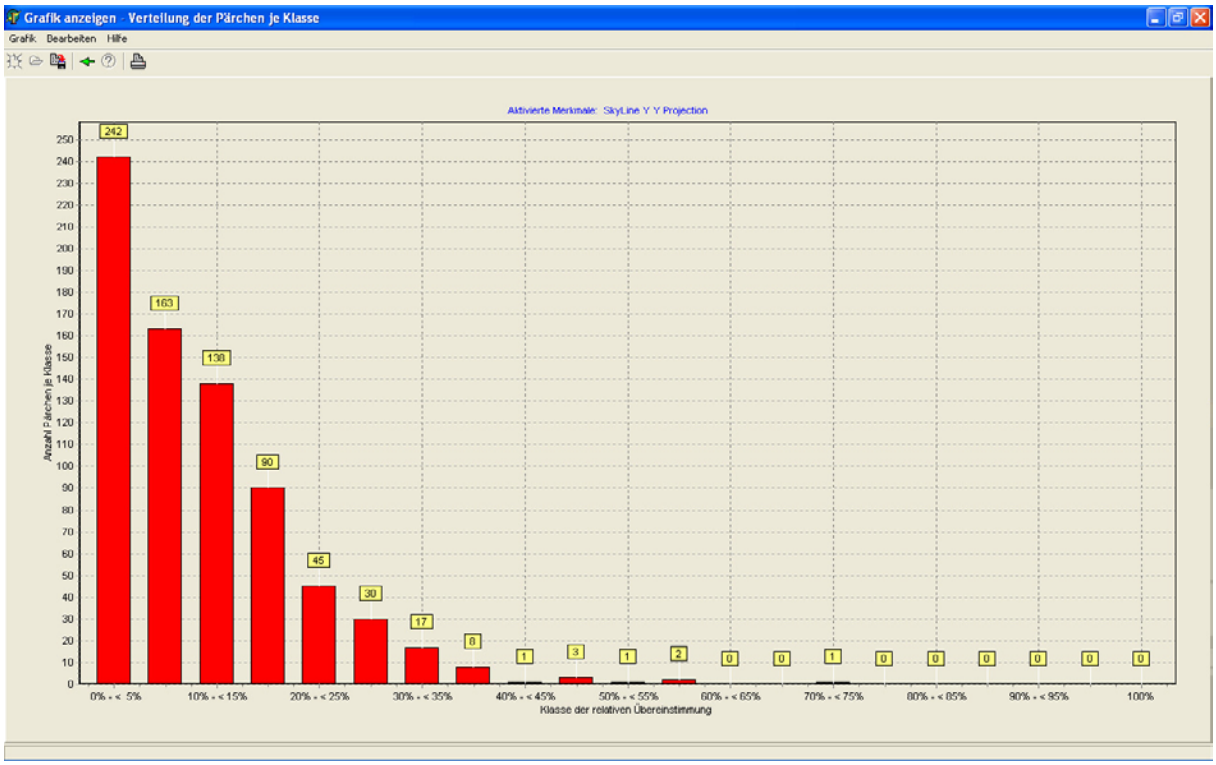


Abbildung 13 Verteilung der relativen Übereinstimmung der SkyLineY und der Y-Projection

Wie die Graphik zeigt, werden die Zeichen in dieser Kombination sehr gut diskriminiert. Es gibt lediglich ein Paar (UÜ), dessen Übereinstimmung zwischen 70% und 75% liegt; die anderen liegen deutlich unter 60%.

2.10 Erfahrungen aus der Praxis

2.10.1 Einleitung

Die o.g. Tabellen und Graphiken deuten darauf hin, dass die SkyLineY und die Y-Projection (einzeln und in Kombination) die Merkmale sind, die die Zeichen am besten diskriminieren können. Die Praxis beim Einsatz der Software bestätigt dies allerdings **nicht** so deutlich. Dort hat sich das Gitter bzw. der Pixelvergleich als das „bessere“ Merkmal herausgestellt; besser im Sinne von besserer Erkennung.

Dies sei an einem Kennzeichen erläutert, das bei der Aktivierung der Merkmale SkyLineY und Y-Projection nicht korrekt erkannt wird.

Abbildung 14 Fehlerhafte Erkennung des Kennzeichens

Es wird statt der „9“ ein „D“, statt der „6“ ein „B“ und statt der „7“ eine „1“ erkannt. Eine Ähnlichkeit der tatsächlich gefundenen Zeichen mit den Erwarteten ist nicht erkennbar. Durch Doppelklick z.B. auf der „9“ kann man das segmentierte Zeichen in eine Bildschirmmaske übernehmen und mit den gelernten Referenzen vergleichen. Dazu wird das segmentierte Zeichen im linken Teil der Maske dargestellt.

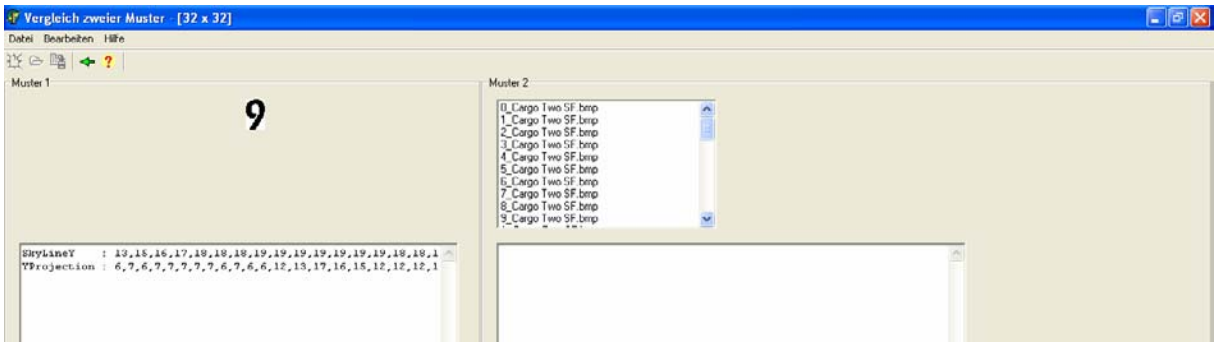


Abbildung 15 Übernahme des segmentierten Zeichens

Die Vektoren der aktivierten Merkmale werden angezeigt. Nun kann man im rechten Teil die „gelernte“ 9 anwählen und anschließend die Merkmalsvektoren vergleichen.

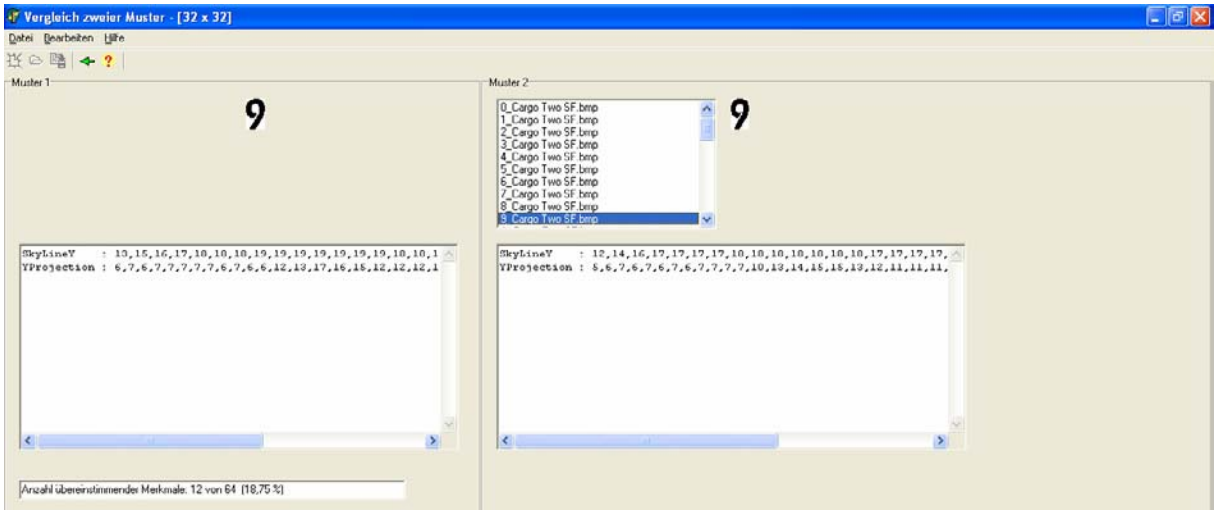


Abbildung 16 Gelerntes Muster im rechten Teil des Bildschirms

Die Merkmalswerte, der besseren Lesbarkeit wegen, herauskopiert und paarweise untereinander geschrieben:

SkyLineY	13,15,16,17,18,18,18,19,19,19,19,19,19,19,18,18,18,18,17,17,16,15,15,14,14,13,13,12,11,10,10,9
	12,14,16,17,17,17,17,18,18,18,18,18,18,18,17,17,17,17,16,16,15,14,14,13,12,12,11,11,10,10,9,8
Y-Projection	6,7,6,7,7,7,7,7,6,7,6,6,12,13,17,16,15,12,12,12,12,12,12,12,13,12,16,18,16,13,11,6
	5,6,7,6,7,6,7,6,7,7,7,7,10,13,14,15,15,13,12,11,11,11,11,11,12,12,17,17,16,14,11,7

Tabelle 8 Die Merkmalsvektoren der zu vergleichenden Zeichen

Dabei wird deutlich, das sich die Merkmalswerte komponentenweise teilweise lediglich um absolut 1 oder 2 unterscheiden. Bei der Ermittlung der relativen Übereinstimmung (bzw. bei der Anfrage an die SpaCAM) wird aber auf **exakte** Übereinstimmung der einzelnen Merkmalskomponenten geprüft.

In diesem Fall stimmen von den 64 Merkmalswerten lediglich 12 **exakt** überein. Somit ergibt sich die relative Übereinstimmung von nur 18,75%. Beim Vergleich der „9“ mit dem „D“ stimmen allerdings 16 Merkmalswerte exakt überein; deshalb beträgt die Übereinstimmung 25% und es wird statt der „9“ das „D“ erkannt. Analog lässt sich dieses für die anderen Zeichen durchführen.

Nun soll analysiert werden, was zur fehlerhaften Erkennung führt.

2.10.2 Gründe für die fehlerhafte Erkennung

Vergrößert man die Bitmaps der „**segmentierten**“ 9 und die der „**gelernten**“ 9 und „legt“ diese beiden Bilder mittels einer XOR-Verknüpfung übereinander, so erkennt man die Ursache:

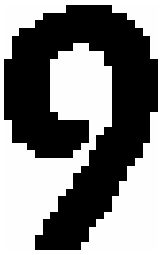


Abbildung 17 „Segmentierte“ 9

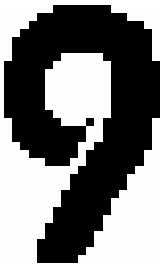


Abbildung 18 „Gelernte“ 9

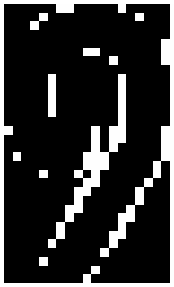


Abbildung 19 XOR-Verknüpfung beider Bilder

Bei der XOR-Verknüpfung werden paarweise gleiche Pixel (also schwarz/schwarz oder weiß/weiß) zu schwarz; die anderen zu weiß. Sind zwei Bilder **exakt** gleich, so wird daraus ein vollständig schwarzes Rechteck. Weiße Pixel signalisieren also pixelweise Unterschiede in den beiden Bildern. Für die SkyLineY sieht man, dass am Rand der 9 eben die Bilder der **segmentierten** und der **gelernten** 9 **nicht** exakt übereinstimmen und zu den oben beschriebenen **geringfügigen** Differenzen in den Merkmalswerten führen.

Das gleiche gilt auch für die XOR-Verknüpfung der Y-Projection, wo es ebenfalls winzige Differenzen gibt:

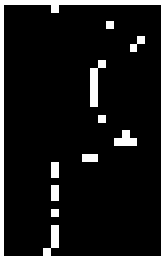


Abbildung 20 XOR-Verknüpfung der Y-Projection

Da die **segmentierten** Zeichen immer optischen Einflüssen wie Unschärfe, Verschmutzung etc. ausgesetzt sind, lassen sich beim **Vergleich** mit den aus Fonts gelernten „astreinen“ Referenzen solche „Pixelfehler“ wohl nicht vermeiden. Ein weiterer Effekt dürfte daraus resultieren, dass die Referenzen immer mit einer **festen** Auflösung (hier 32x32) gelernt wurden. Die **segmentierten** Zeichen müssen ggf. durch Vergrößerung oder Verkleinerung auf dieses Referenzmaß normiert werden, woraus sich auch mehr oder weniger große Abweichungen ergeben.

Bei einer Auflösung von 32x32 haben die Merkmalsvektoren der SkyLine, Projection, GroundLine etc. lediglich jeweils 32 Komponenten. Ein einziger Pixelfehler reduziert damit die Übereinstimmung um ca. 3% je Vektor.

Hierin scheint der „Erfolg“ des Merkmals Pixelvergleich zu liegen. Ein Pixelfehler am Rand der SkyLine bzw. ein Pixelfehler innerhalb oder auf dem Rand des Musters

verändert **mindestens** eine Komponente des zugehörigen Merkmalsvektors. Dies ist zwar bei dem Merkmal Pixelvergleich nicht anders, tritt aber nicht so signifikant in Erscheinung. Bei einer Auflösung von 32x32 gibt es insgesamt 1024 Pixel (und damit 1024 Komponenten im Merkmalsvektor), die gesetzt oder nicht gesetzt sein können. Ein Pixelfehler reduziert damit die Übereinstimmung lediglich im Promillebereich.

Durch Verschmutzung, Unschärfen, Verzerrungen etc. sind insb. die Ränder der segmentierten Zeichen nicht glatt sondern eher „ausgefranst"; d.h. einzelne Pixel fehlen, dafür gibt es an anderen Stellen mehr. Bei der SkyLineY ist aber gerade der **rechte Rand** des Musters entscheidend und das „Ausfransen" ist verantwortlich für die Differenzen in den Komponentenwerten. Bereits wenige solcher Pixelfehler genügen, die relative Übereinstimmung so zu verschlechtern, dass ein gänzlich „unähnliches" Zeichen als das „Beste" erkannt wird. Gleiches lässt sich für die Y-Projection herleiten, da zuviel gesetzte Pixel die Komponentenwerte erhöhen und fehlende Pixel entsprechend erniedrigen können. Die gleiche Argumentation lässt sich auch für die X-Projektion, die SkyLineX und die GroundLine führen.

Wie es scheint, ist die **Häufung** von Pixeln in bestimmten Regionen ausschlaggebend für die Erkennung. Das Merkmal für die Häufung von Pixeln in bestimmten Regionen ist das **Gitter**, bzw. der Pixelvergleich als Spezialfall des Gitters. Das Merkmal des Pixelvergleichs reagiert hier deutlich unempfindlicher, da einige Pixel mehr oder weniger die **Gesamtheit aller Pixel** (gesetzt oder nicht gesetzt) nur unwesentlich verändert.

Bei der spärlichen Codierung sind die gewählten Parameter des Gitters (Anzahl der Spalten, Anzahl der Zeilen) ausschlaggebend für die Auslastung der SpaCAM-Vektoren bzw. der SpaCAM-Matrix. Dies soll nun näher untersucht werden.

2.10.3 Auswirkungen auf die SpaCAM

Das Merkmal Gitter soll bei einem auf 32x32 Pixel normierten Muster mit verschiedenen Parametern spärlich codiert werden.

Bei einem Gitter mit 8 Spalten und 8 Zeilen entstehen $8 \cdot 8 = 64$ Felder mit jeweils $4 \cdot 4 = 16$ Pixeln. Der (dezimale) Merkmalsvektor hat dann 64 Komponenten, wobei jede Komponente Werte von 0 bis 16 annehmen kann. Wird dieser Vektor spärlich codiert, so hat der SpaCAM-Vektor eine Länge von $64 \cdot 17 = 1088$ Bits, wobei immer 64 Bits gesetzt sind, während die restlichen Null sind. Daraus ergibt sich eine Auslastung des SpaCAM-Vektors (bzw. der SpaCAM-Matrix) von $\frac{64}{1088} \approx 5{,}8\%$.

In einem Gitter mit 16 Spalten und 16 Zeilen entstehen $16 \cdot 16 = 256$ Felder mit jeweils 4 Pixeln. Der SpaCAM-Vektor hat dann $256 \cdot 5 = 1280$ mit 256 Einsen. Daraus resultiert eine Auslastung von 20%.

Entspricht im Gitter die Anzahl der Spalten der Breite des Musters und die Anzahl der Zeilen der Höhe des Musters, so entstehen $32 \cdot 32 = 1024$ Felder mit jeweils 1 Pixel. Der SpaCAM-Vektor hat dann die aus dem **Pixelvergleich** bekannten $1024 \cdot 2 = 2048$ Bits mit 1024 gesetzten Einsen. Die Auslastung der SpaCAM-Matrix ist dann 50%.

Das bedeutet, dass das Gitter mit zunehmender „Dichte“ die Auslastung der SpaCAM-Maxtrix bis zu 50% verschlechtert, nämlich auf die Auslastung des Merkmals des Pixelvergleichs.

In der Praxis wird man versuchen, die Parameter für das Gitter soweit anzupassen, dass optimale Erkennungsergebnisse erreicht werden; im Grenzfall kann dies natürlich auch der Pixelvergleich sein.

2.11 Anfragen

Nach dem die SpaCAM mit den gewünschten Merkmalen gelernt wurde, können nun Anfragen erfolgen. Nach der Segmentierung des Ausgangsbildes und der Merkmalscodierung der normierten Segmente kann für jedes Segment eine Anfrage an die SpaCAM, wie oben dargelegt, erfolgen. Man bekommt damit für jedes Segment die

abgestufte absolute und relative Übereinstimmung mit allen gelernten Referenzen. Hier kann jetzt, wenn gewünscht, eine weitere Filtermöglichkeit eingesetzt werden:

(1) Wird eine Mindestübereinstimmung (absolut oder relativ) definiert, so können alle gefundenen Referenzen, die unter dieser Grenze liegen, eliminiert werden.

(2) Wird eine Trefferanzahl x definiert, so werden nur die x „besten“ Referenzen weiterhin betrachtet.

Da die Referenzen „wissen“, welchen Buchstaben oder welche Ziffer sie repräsentieren, kann daraus der String des erkannten Autokennzeichens ermittelt werden.

2.12 Ermittlung des Kfz-Kennzeichens

Beispiel:

Die Anfrage für 3 Segmente hat folgende 2 Referenzen als „beste“ Treffer ermittelt:

Segment 1

„K“ 80%

„X“ 75%

Segment 2

“8” 85%

“B” 84%

Segment 3

“A” 80%

“H” 75%

Daraus lassen sich nun $2^3 = 8$ mögliche Strings kombinieren und als Mittelwert die relative Übereinstimmung des Gesamtstrings ermitteln:

String	Güte	
K8A	(80+85+80)/3=	81,7
K8W	(80+85+75)/3=	80,0
KBA	(80+84+80)/3=	81,3
KBW	(80+84+75)/3=	79,9
X8A	(75+85+80)/3=	80,0
X8W	(75+85+75)/3=	76,7
XBA	(75+84+80)/3=	79,7
XBW	(75+84+75)/3=	78,0

Sortiert man diese Tabelle absteigend nach den Güten, so erhält man eine Abstufung der „besten“ Kfz-Kennzeichen. In diesem Fall ist K8A aufgrund des Mittelwerts der Einzelgüten, der „beste“ Treffer.

String	Güte	
K8A	(80+85+80)/3=	81,7
KBA	(80+84+80)/3=	81,3
K8W	(80+85+75)/3=	80,0
X8A	(75+85+80)/3=	80,0
KBW	(80+84+75)/3=	79,9
XBA	(75+84+80)/3=	79,7
XBW	(75+84+75)/3=	78,0
X8W	(75+85+75)/3=	76,7

Wie man sieht, wächst die Anzahl möglicher Kombinationen exponentiell mit der Anzahl gewünschter Treffer. Hier ist also Vorsicht geboten, um nicht zu lange Antwortzeiten zu bekommen.

Hieran kann sich nun eine semantische Prüfung des gefundenen Kennzeichens anschließen, denn das erste Zeichen muss ein Buchstabe sein und nach einer Ziffer

darf kein Buchstabe mehr kommen (zumindest in Deutschland; abgesehen von dem abschließenden „H“ bei Oldtimern).

Nun liegen, je nach gewünschter Trefferzahl, eine Anzahl von Strings vor, die das erkannte Kfz-Kennzeichen enthalten (können). Diese Strings sollen nun als Schlüssel für eine Datenbank dienen, um nach weiteren Angaben zu suchen.

2.13 Suchen in einer Datenbank mit der SpaCAM-Technik

2.13.1 Einleitung

Die Datenbank soll, hier stark vereinfacht dargestellt, in folgender Form vorliegen:
Es gibt eine Relation mit dem String-Feld „Kfz-Kennzeichen“ und beliebig vielen weiteren Feldern mit den Sekundärinformationen, zusammengefasst unter „Diverses“.

Kfz-Kennzeichen	Diverses
KBAP150	...
GÖXY220	...
...	...

Da die Kennzeichenerkennungs-Strings in der Datenbank fehlertolerant gefunden werden sollen, bietet sich folgender Lösungsweg an:

(1) Lernen der gespeicherten Kfz-Kennzeichen in einer SpaCAM
(2) Anfragen der Strings aus der Kennzeichenerkennung an die SpaCAM, um so das (oder die) „beste“ Übereinstimmung des/der erkannten Kennzeichen(s) mit den Kennzeichen in der Datenbank zu bekommen.

2.13.2 Merkmalscodierung

Die Technik der SpaCAM bleibt wie gehabt. Man muss sich lediglich überlegen, wie man die Merkmalsextraktion und –codierung umsetzt.

Für die Texterkennung (Textretrieval) wird folgende Codierung eingesetzt [**HAG1996**]:

Man bildet Paare aller möglichen Zeichenkombinationen, die im Ein- und Ausgabevektor vorkommen können. Um auch Textanfang und Textende erkennen

zu können, setzt man ein Zeichen ein, das für die Texterkennung keine Rolle spielt. So kann z.B. für Textanfang und –ende das Zeichen „#“ gewählt werden.

Da für die Autokennzeichen in Deutschland lediglich die 26 Großbuchstaben, 3 Umlaute und 10 Ziffern vorkommen können, keine Ziffern am Anfang und keine Buchstaben (abgesehen vom „H“ für Oldtimer) am Ende stehen, gibt es folgende Paare (Auszug):

#A bis #Z
AA bis AZ
A0 bis A9
AÄ bis AÜ
...
BA bis BZ
B0 bis B9
BÄ bis BÜ
...
ÄA bis ÄZ
...
00 bis 09
10 bis 19
...
90 bis 99
0# bis 9#

Diese Paare kann man in einer Tabelle darstellen:

#A	..	#Z	AA	..	AZ	A0	..	A9	AÄ	..	AÜ	…		0#	..	9#

Zerlegt man nun jedes Kfz-Kennzeichen aus der Datenbank in “seine” Paare und setzt **zeilenweise** für jedes Kfz-Kennzeichen aus der Datenbank eine 1 für ein vor-

handenes und eine 0 für ein nicht vorhandenes Paar, so erhält man wiederum eine – sicherlich spärlich – codierte Assoziativmatrix aller gespeicherten Kfz-Kennzeichen.

Beispiele:

- Für KBAP150 wird bei #K, KB, BA, AP, P1, 15, 50 und 0# eine 1 gesetzt und sonst eine 0.
- GÖXY220 bekommt bei #G, GÖ, ÖX, XY, Y2, 22, 20 und 0# eine 1 und sonst eine 0.

„Zerlegt" man die Anfragevektoren aus der Kennzeichenerkennung nach den gleichen Regeln kann man durch „Matrixmultiplikation" mit der SpaCAM wiederum die Übereinstimmungen der Anfragevektoren mit den gelernten Kfz-Kennzeichen erhalten. Sollte das Kfz-Kennzeichen exakt erkannt werden und in der Datenbank enthalten sein, so wird es mit Sicherheit gefunden. Aufgrund der fehlertoleranten SpaCAM Technik werden aber auch „verstümmelte" oder nicht komplett erkannte Autokennzeichen in der Datenbank gefunden. Definiert man auch hier eine Mindestgüte, so kann man weiterhin auch feststellen, ob das erkannte Kennzeichen überhaupt in der Datenbank ist bzw. sein kann.

Mit dieser Technik kann man für alle erkannten Kfz-Kennzeichen die wiederum „Besten" oder „Passendsten" in der Datenbank finden. Mit dem Finden in der Datenbank hat man aber den Primärschlüssel, über den dann, mit herkömmlichen Datenbanktechniken, beliebige weitere Auswertungen erfolgen können.

Diese Technik lässt sich natürlich von Zeichenpaare auf 3er-, 4er- etc. Gruppen erweitern [**HAG1996**]. Sollten die Zeichengruppen mehrfach im Text vorkommen, so werden sie allerdings nur einmal codiert. Somit eignet sich diese Technik nur für nicht allzu lange Texte, da dort die Wahrscheinlichkeit des mehrfachen Vorkommens von Zeichengruppen geringer ist.

In der Applikation sind die Funktionen bereits implementiert, um die Strings der erkannten Kennzeichen nach o.g. Verfahren spärlich zu codieren und für eine mögliche Weiterentwicklung der Software zur Verfügung zu stellen.

2.13.3 Algorithmus

Der **allgemeine** Algorithmus[5] zur Generierung eines spärlich codierten Binärvektors V aus einem beliebigen String S aus der Menge A an „erlaubten“ Zeichen und dem Textanfang- bzw. Textendezeichen # unter Berücksichtigung von Zeichengruppen der Länge n ist recht einfach.

Zunächst generiert man **rekursiv** alle möglichen Zeichengruppen der Länge n aus A+{#} (dies sind $|A+\{\#\}|^n$) und speichert diese in einer sortieren Liste L. Anschl. fügt man dem String S am Anfang und Ende das Zeichen # zu.

Dann ermittelt man alle Zeichengruppen der Länge des Strings S von links an und ermittelt den Index dieser Zeichengruppe in der Liste L. Das dem Index entsprechende Bit im Vektor V wird dann gesetzt.

Der Aufbau der Liste und damit die Größe des Vektors V hat eine Komplexität von $O(|A+\{\#\}|^n)$. Das Suchen der Zeichenfolgen in der sortierten Liste eine Komplexität von $O(lb(|A+\{\#\}|^n)) = O(n \cdot lb(|A+\{\#\}|))$. Somit sollte man mit dem n vorsichtig umgehen.

Wenn, wie für diese Arbeit ausreichend, A = {Großbuchstaben, Ziffern und Umlaute} ist, so ist die Kardinalität |A|=39. Bei Zahlenpaaren (n=2) ergibt sich dann eine Liste mit insgesamt $39^2 = 1521$ Einträgen. Dies ist auch die Länge des SpaCAM-Vektors.

[5] ohne semantische Prüfung des Kennzeichens auf z.B. Buchstaben am Anfang etc.

2.14 Weitere Anwendungsgebiete der SpaCAM

Die SpaCAM lässt sich nicht nur für die Erkennung von Unterschriften und Kfz-Kennzeichen, sondern auch für eine Vielzahl anderer Aufgaben in der Informatik einsetzen [**HEIT1994**]:

- Dublettensuche und Sortieren
- Darstellung von Mengen
- Komprimierungstechniken
- Textmustersuche

Aber auch in klassischen Datenbankanwendungen, im Informations-Retrieval, zur Homologie-Suche in DNA (Primärstruktur) und die fehlertolerante Suche in Objekten mit unendlichem Informationsgehalt, z.B. durch die Zeichenfolgen der DNA, kommt diese Technik zum Einsatz [**LUBA2001**].

Bei diesen Untersuchungen hat sich gezeigt, dass die SpaCAM-Technik oft effizienter arbeitet als vergleichbare konventionelle Verfahren.

2.15 Alternative Ansätze

An der Universität von Kuala Lumpur **[YAP]** wurde ein auf das malaysische Autokennzeichen abgestimmtes Verfahren zur Kennzeichenerkennung entwickelt. Nach der Segmentierung der Zeichen werden zur Erkennung **Neuronale Fuzzy Netze** genutzt. (Zu Fuzzy Netzen siehe auch [**PETRY1999**]. Allgemeines zur Fuzzy-Logik siehe auch [**HAG1996**]).

[WEV1998] beschreibt ein Verfahren zur Segmentierung von Schriftzeichen und Symbolen auf Kfz-Kennzeichenschildern, bei dem nach zusammenhängenden Strukturen von Pixeln vorgegebener geometrischer Eigenschaften (sog. Blobs) gesucht wird. Die so gefundenen Strukturen mit ihren Koordinaten werden in einer Konnektivitätsanalyse auf gegenseitige Position geprüft. Werden dabei vorgegebenen Formatvorschriften erfüllt, so werden diese Strukturen einzeln in geordneter Reihenfolge

zur weiteren Bearbeitung (z.B. zur Zeichendekodierung, Mustererkennung etc.) weitergeleitet.

[**SCHÜ1977**] beschreibt ein gänzlich anderes, auf der **Nachrichtentechnik** basierendes stochastisches Verfahren zur Erkennung von Zeichen. Dieses soll in seinen **Grundzügen kurz** beschrieben werden.

Der Autor geht von einem Zeichenerzeugenden Prozess $\{V,k\}$ aus, wobei V den Merkmalsvektor des Zeichens – z.B. die geschwärzte Rasterung eines OCR-A Zeichens – und k dessen Klassenindex darstellt. Bei K **unterschiedlichen** zu erkennender Zeichen ist $k \in \{1,2,...,K\}$ [6]. Das Erkennungssystem muss nun das tatsächlich gesendete und zu erkennende Zeichen ϖ, von dem lediglich V existiert, möglichst zuverlässig in eine der Klassen k = 1,2,...,K einordnen. Sollte ϖ nicht oder nicht hinreichend erkannt werden können, so wird es in die Klasse $k = 0$, die sog. Abweisungsklasse, eingeordnet. Folglich gilt $k \in \{0,1,...,K\}$.

Da das gleiche Zeichen in verschiedenen Ausprägungen des Merkmalsvektors V gezeichnet werden kann[7], erzeugt der Zeichenerzeugende Prozess $\{V,k\}$ Wertepaare $[V,k]$. Dazu wird $\{V,k\}$ als Zufallsprozess betrachtet, da die Wertepaare $[V,k]$ zwar zufällig, aber nicht regellos entstehen. Der Prozess $\{V,k\}$ wird regiert von einem Verteilungsgesetz $p(V,k)$, das entweder a priori bekannt bzw. gegeben ist oder durch eine genügend lange Beobachtungsdauer approximativ ermittelt werden kann. Mit Hilfe der bedingten Wahrscheinlichkeit wird das Verteilungsgesetz zu:

$$p(V,k) = p(V \mid k) \cdot p(k)$$

wobei $p(k)$ als die Quellenstatistik und $p(V \mid k)$ als klassenspezifische Verteilung der Merkmalsvektoren V, die der Klasse k angehören, bezeichnet werden.

Die Erkennung von ϖ soll nun mit Hilfe des entscheidungstheoretischen Lösungsansatzes erfolgen. Dazu wird eine **deterministische** Entscheidungsfunktion e(V) mit

[6] Sollen z.B. die Ziffern 0 bis 9 erkannt werden, so ist K = 10; bei den Großbuchstaben ohne Umlaute entsprechend K = 26 etc.

[7] z.B. durch Unschärfe, Rauschen, Drehung etc.

dem Wertevorrat der K+1 Klassen, also $e \in \{0,1,....,K\}$, eingeführt. Deterministisch deshalb, damit aus dem selben Merkmalsvektor V auch immer die selbe Entscheidung e erzeugt wird. Mit Hilfe dieser Entscheidungsfunktion wird der aus dem tatsächlich gesendeten Zeichen ϖ der Klasse k erzeugten Merkmalsvektor V ein rekonstruierter Klassenindex $\hat{k}$ gebildet. Ist die getroffene Entscheidung $\hat{k}$ richtig, so stimmen k und $\hat{k}$ überein. Sie ist falsch, wenn $\hat{k} \neq 0$ ist und nicht mit k übereinstimmt. Die Entscheidung wird verweigert, wenn $\hat{k} = 0$ ist.

Zur Optimierung des Systems wird nun eine Kostenfunktion c= $C(k,\hat{k})$ eingeführt, die jeder möglichen Konstellation von der Entscheidung $\hat{k}$ und dem Klassenindex k des tatsächlich gesendeten Zeichens einen Kostenwert zuweist. Durch die beiden **deterministischen** Abbildungen $\hat{k} = e(V)$ und $c = C(k,\hat{k})$ entsteht ein stochastischer Prozess {c} schwankender Kosten. Über diesen kann man den Erwartungswert, der auch als Risiko R bezeichnet wird, bilden; d.h. $R = E\{c\} = \sum_V \sum_k C(k, e(V)) \cdot p(V,k)$.

Durch den Einsatz der bedingten Wahrscheinlichkeit ergibt sich $R = \sum_V \sum_k C(k, e(V)) \cdot p(k \mid V) \cdot p(V) = \sum_V R_V(e,V) \cdot p(V)$ wobei

$R_V(e,V) = \sum_k C(k, e(V) \cdot p(k \mid V)$ als das **bedingte Risiko** bezeichnet wird, das dann eintritt, wenn V vorliegt und die Entscheidungsregel e(V) angewandt wird.

Das Risiko R soll minimiert werden. Da ohne Einschränkung der Allgemeinheit angenommen werden kann, dass die Kostenfunktion nur nichtnegative Werte annimmt, ist auch das bedingte Risiko eine nicht negative Größe. Das Risiko R, als Summe nichtnegativer Summanden, ist aber nur dann minimal, wenn sämtliche Summanden minimal sind. Dies bedeutet aber, dass alle **bedingten** Risiken minimiert werden müssen. Folglich muss man lediglich für jeden möglichen Wert von V, unabhängig von den anderen, die Entscheidungsregel optimieren. Für jedes V entsteht dann eine eigene Entscheidungsvorschrift e(V).

Zur Erkennung von ϖ betrachtet man dann genau einen der möglichen Werte von V. Für diesen einen Wert ist die Entscheidung e gesucht, die das bedingte Risiko minimiert. Da e gemäß Definition aber lediglich K+1 unterschiedliche Werte annehmen kann, führt man einfach sämtliche mögliche Entscheidungen $\tilde{e}$ durch und erhält K+1

Maßzahlen. Die kleinste dieser Maßzahlen ist dann das minimale Risiko und gibt an, welche Entscheidung in der betrachteten Situation die Optimale ist.

Der Algorithmus ergibt sich in diesem speziellen Fall zu:

- Berechne für alle K+1 mögliche Entscheidungen $\tilde{e}$

  $$R(\tilde{e}) = \sum_{k=1}^{K} C(k,\tilde{e}) \cdot p(k \mid V)$$

- suche unter allen $R(\tilde{e})$ den kleinsten Wert $R(e)$
- gib die so gefundene Entscheidung e als $\hat{k}$ aus.

Wie man sieht, genügen allein die **Rückschlusswahrscheinlichkeiten** $p(k \mid V)$, an Stelle des vollständigen Verteilungsgesetzes $p(V,k)$. Ist aber das vollständige Verteilungsgesetz gegeben, so können die Rückschlusswahrscheinlichkeiten einfach aus

$p(k \mid V) = \frac{p(V,k)}{p(V)}$ mit der Randverteilung $p(V) = \sum_{k=1}^{K} p(V,k)$ berechnet werden.

Als einfache symmetrische Kostenfunktion kann man die wählen, bei der alle Zurückweisungen mit einen Kostenwert $C_r > 0$ und alle Falschbewertungen einheitlich mit einem anderen Kostenwert $C_f > 0$ bewertet werden. Die korrekte Entscheidung erzeugt dabei keine Kosten. Daraus ergibt sich:

$$C(k,\hat{k}) = \begin{cases} C_r & \text{für } \hat{k} = 0 \\ 0 & \text{für } \hat{k} \neq 0, \hat{k} = k \\ C_f & \text{für } \hat{k} \neq 0, \hat{k} \neq k \end{cases}$$

Der eigentliche Erkennungsalgorithmus hat eine geringe Komplexität. Allerdings ist es recht aufwändig, das vollständige Verteilungsgesetz hinreichend genau anzugeben, insb. dann, wenn je Klasse ein anderes Verteilungsgesetz gilt und/oder nicht immer die Normalverteilung angenommen werden kann. Diesen und ähnlichen Fragen widmet sich [**SCHÜ1977**] intensiv.

Kommerzielle Lösungen finden sich z.B. bei **[AXXTEQ], [CC2000], [ELTEC_a], [ELTEC_b], [POLYSCAN] oder [TRAFFICSCAN]**.

---

3 Die Applikation

3.1 Vorbemerkung

Das System arbeitet mit zwei Verzeichnissen:

1. Dem Arbeitsverzeichnis (Daten), in dem die Bilder der zu analysierenden Autokennzeichen enthalten sind.
2. Dem Referenzverzeichnis (Referenzen), in dem die Bilder der Buchstaben und Zahlen von (bereits gelernten) Autokennzeichen enthalten sind. Bei der Installation dieser Software ist dieses Verzeichnis leer. Deshalb sind zunächst einige Vorarbeiten nötig.

3.2 Installationshinweise

Vor der Installation der Software sollte folgendermaßen vorgegangen werden:

- Einrichten eines Verzeichnisses auf der Festplatte (z.B. C:\Mustererkennung).
- Kopieren der Dateien (MUSTERERKENNUNG.*) (von Diskette und/oder CD) in dieses Verzeichnis.
- Ggf. Kopieren und Installieren des Zeichensatzes für die neuen Autokennzeichen (Cargo Two FS).
- Einrichten eines Verzeichnisse für die Bilder (z.B. C:\Mustererkennung\Autokennzeichen).
- Kopieren der Bilder in dieses Verzeichnis; das Programm arbeitet mit allen gängigen Bildformaten (BMP, JPEG etc.)
- Einrichten eines Verzeichnisses für die Referenzen (z.B. C:\Mustererkennung\Referenzen).

Nun kann die Software MUSTERERKENNUNG.EXE z.B. aus dem Explorer heraus gestartet werden. Es erscheint der Startbildschirm.

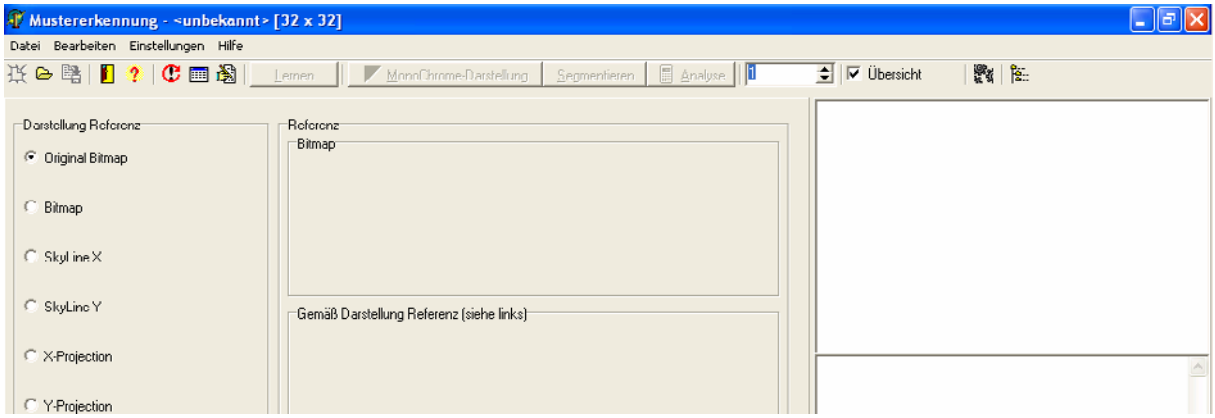


Abbildung 21 Startbildschirm

Bei der **Erstbenutzung** des Programms wird automatisch in den Teil der Optionen (/***Einstellungen/Optionen***) verzweigt, wo die Parameter für die Mustererkennung definiert bzw. geändert werden können (s. unten Kapitel Optionen)

Über das Menü ***/Einstellungen/Customizing*** in das Customizing wechseln.

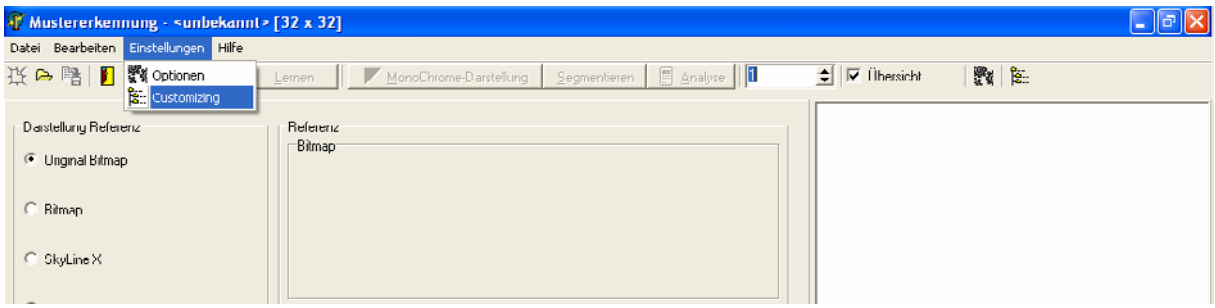


Abbildung 22 Wechseln in das Customizing

Dort ist der Reiter ***Verzeichnisse*** durch Klick auszuwählen. Zunächst stehen alle Verzeichnisse auf dem Verzeichnis, in dem das Programm installiert und gestartet wurde. **Deshalb sollte Home nicht geändert werden**.

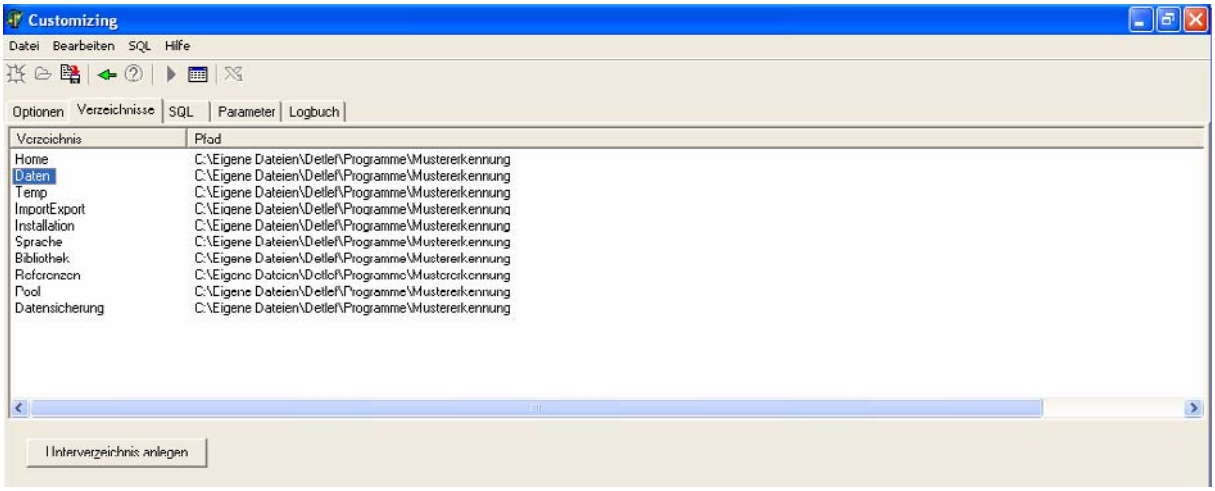


Abbildung 23 Verzeichniseinstellung

Mit einem Doppelklick auf ***Daten*** kann das Verzeichnis für die Autokennzeichen eingerichtet werden.

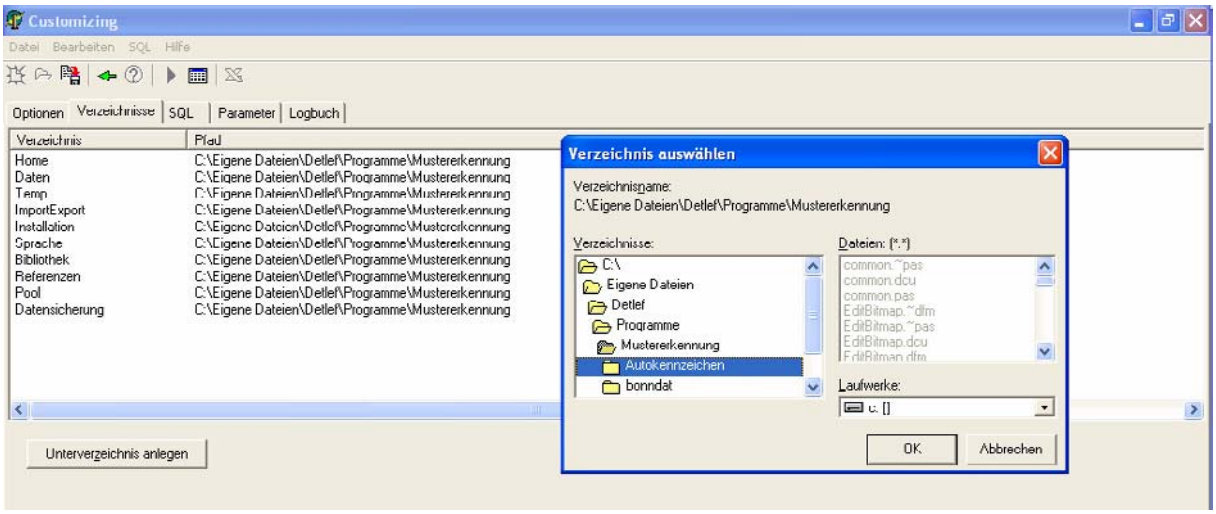


Abbildung 24 Einstellung des Verzeichnisses für die Autokennzeichen

Doppelklick auf ***Referenzen:*** Das Verzeichnis für die Referenzmuster einstellen.

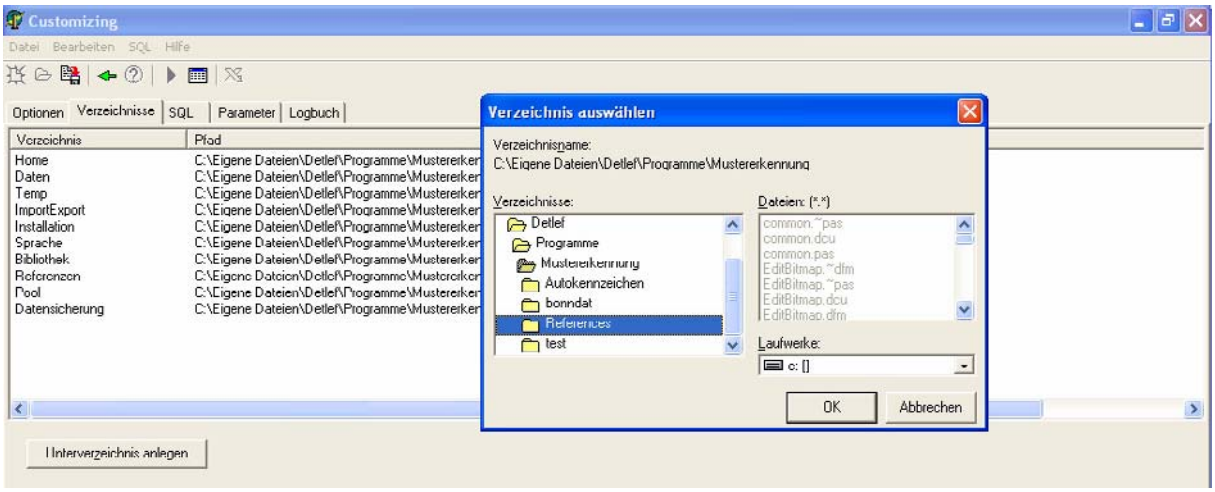


Abbildung 25 Einstellung des Referenzverzeichnisses

Anschließend müssen die Einstellungen gespeichert werden (***/Datei/Speichern***).

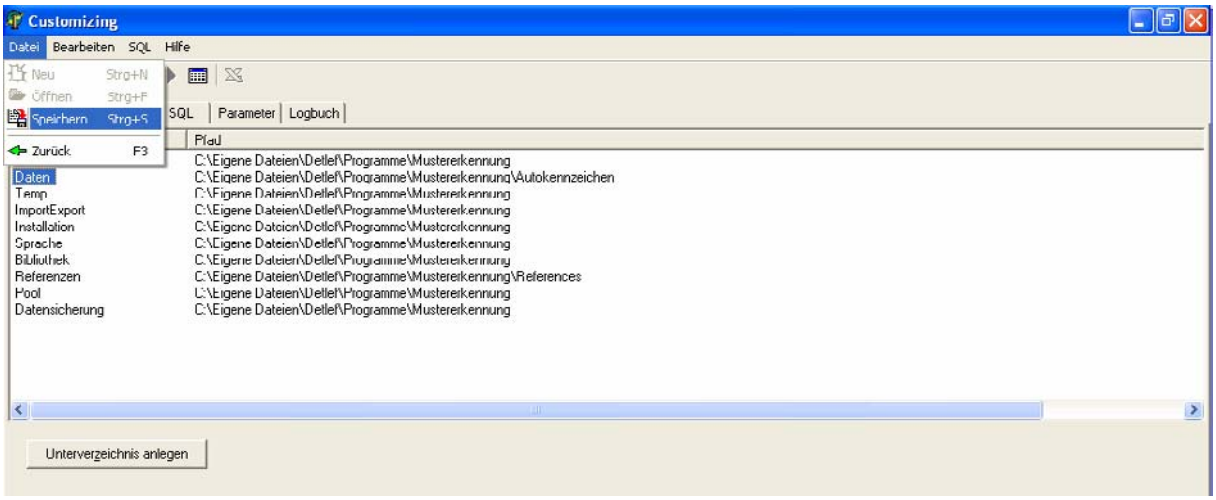


Abbildung 26 Speichern der Einstellungen

Über ***/Datei/Zurück*** wird der Bildschirm verlassen. Wurden Verzeichniseinträge geändert, so wird das Programm beendet. Anschl. kann es erneut mit den neuen Einstellungen gestartet werden.

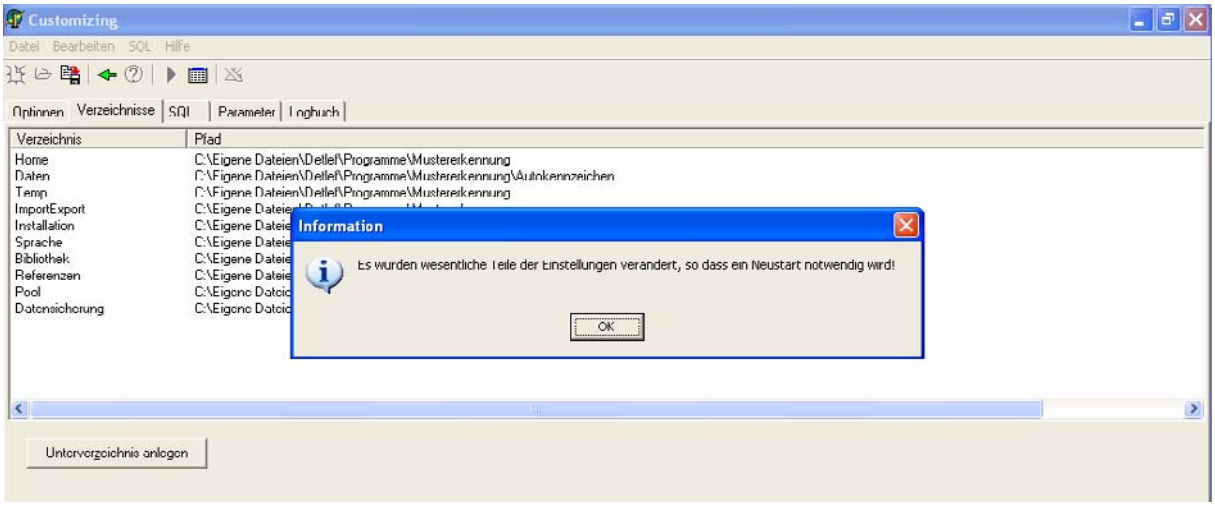


Abbildung 27 Hinweis zum Neustart des Programms

3.3 Startbildschirm

Nach dem (erneuten) Start der Software präsentiert sich der Bildschirm in gewohnter Windows-Struktur:

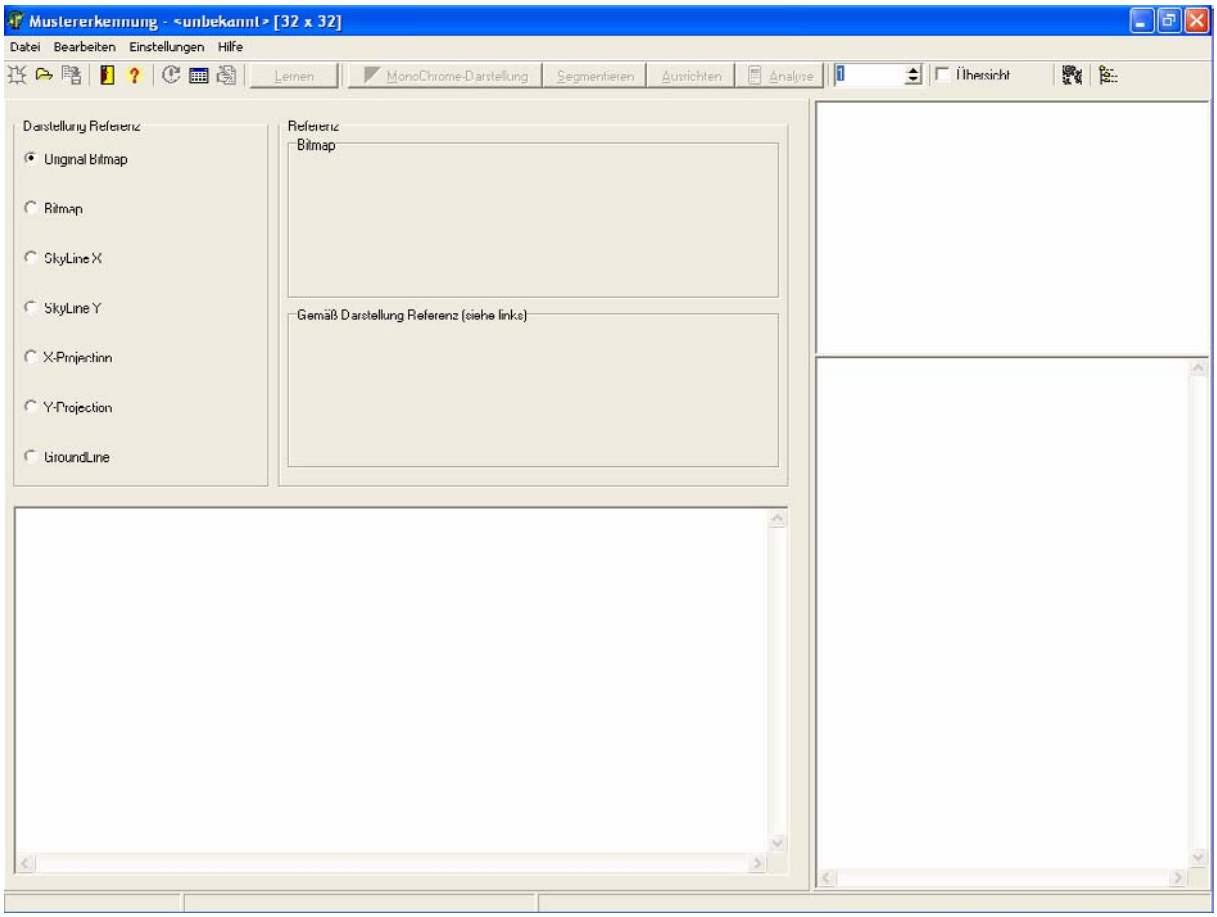


Abbildung 28 Startbildschirm

Über das Menü ***/Datei/Öffnen*** kann hier ein zu analysierendes Foto aus dem im Customizing eingestellten **Daten**verzeichnis geladen werden

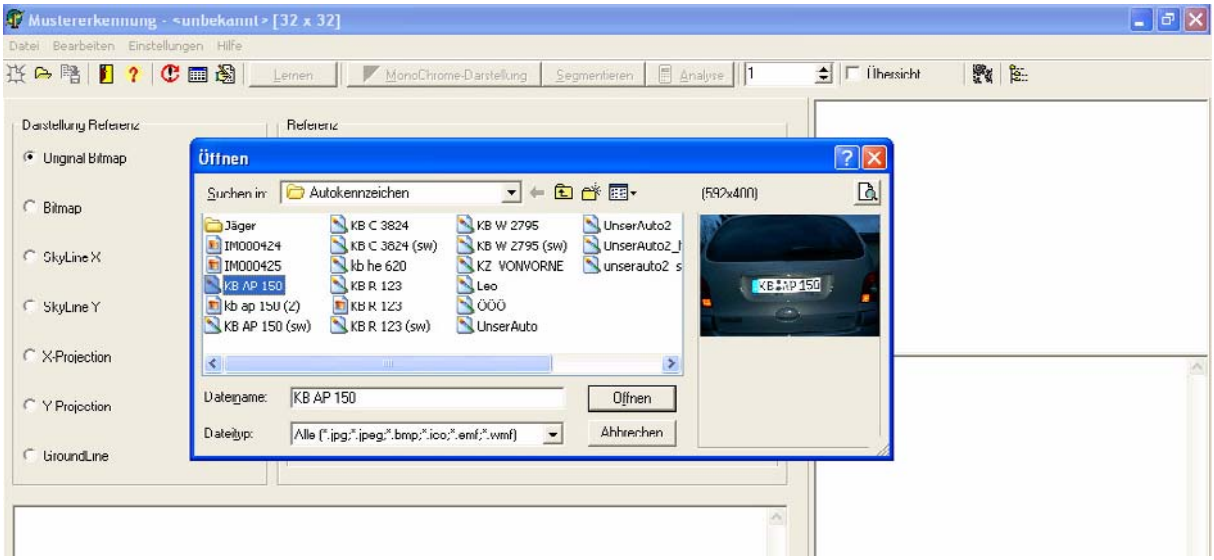


Abbildung 29 Laden eines zu analysierendes Fotos

Das Foto wird in ein eigenes, frei verschiebbares Fenster geladen.

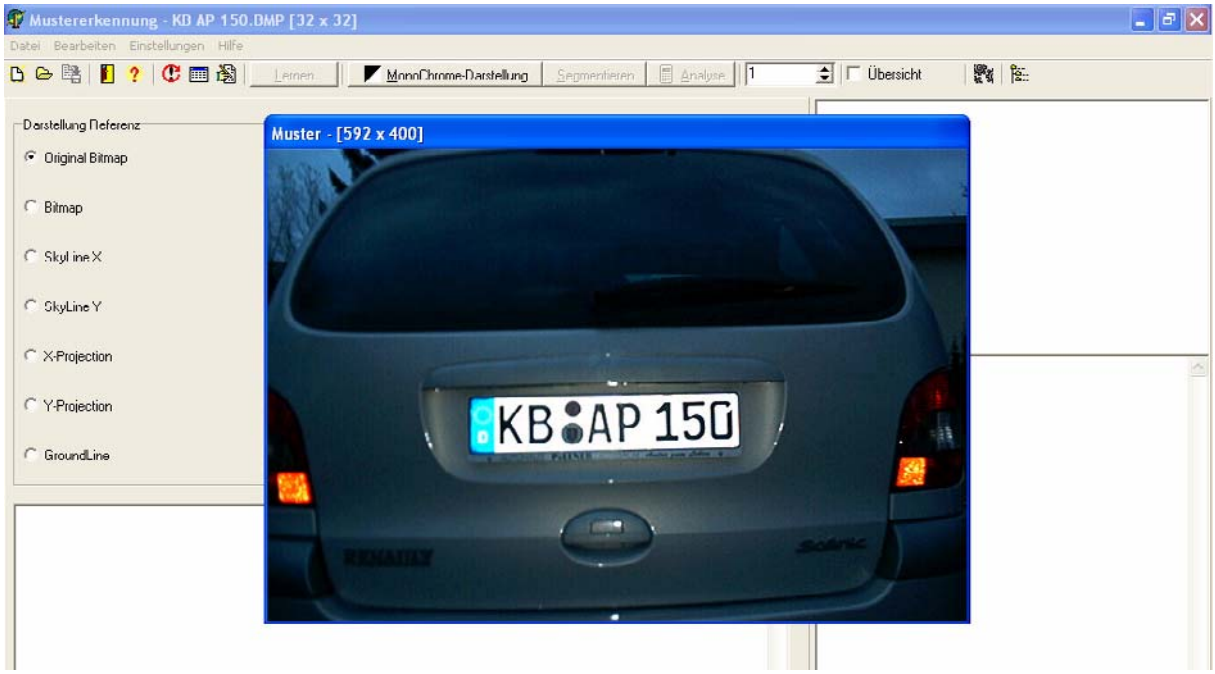


Abbildung 30 Foto ist geladen

Nun könnte man mit einem Klick auf ***Analyse*** den Erkennungsprozess in Gang bringen. Da aber derzeit das Referenzverzeichnis (oben rechts im Bildschirm würden die Referenzmuster angezeigt) leer ist, würde dies nichts bringen. Außerdem ist das Foto noch in Farbe und muss zunächst nach monochrom umgewandelt werden.

Abbildung 31 Das Foto nach der monochromen Umwandlung

Soll z.B. das Foto nach dem Laden automatisch monochrom werden, so kann dies u.a. über ***Einstellungen /Optionen*** im Reiter Monochrome eingestellt werden.

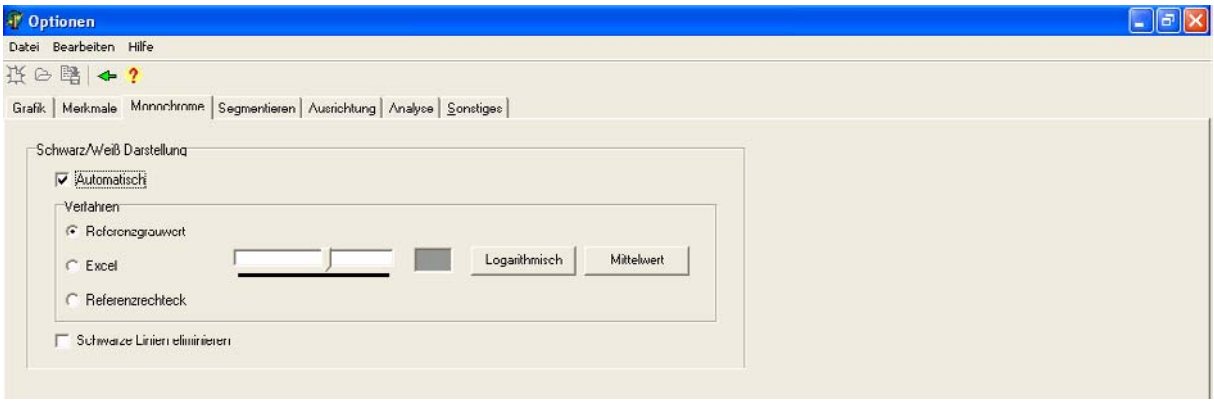


Abbildung 32 Das Foto wird automatisch monochrom

3.4 Segmentieren

Nun muss das Foto in die Buchstaben, Zahlen, etc. segmentiert, d.h. in durch „weiße“ Rechtecke abgegrenzte Bereiche zerlegt werden. (Eigener Button!).

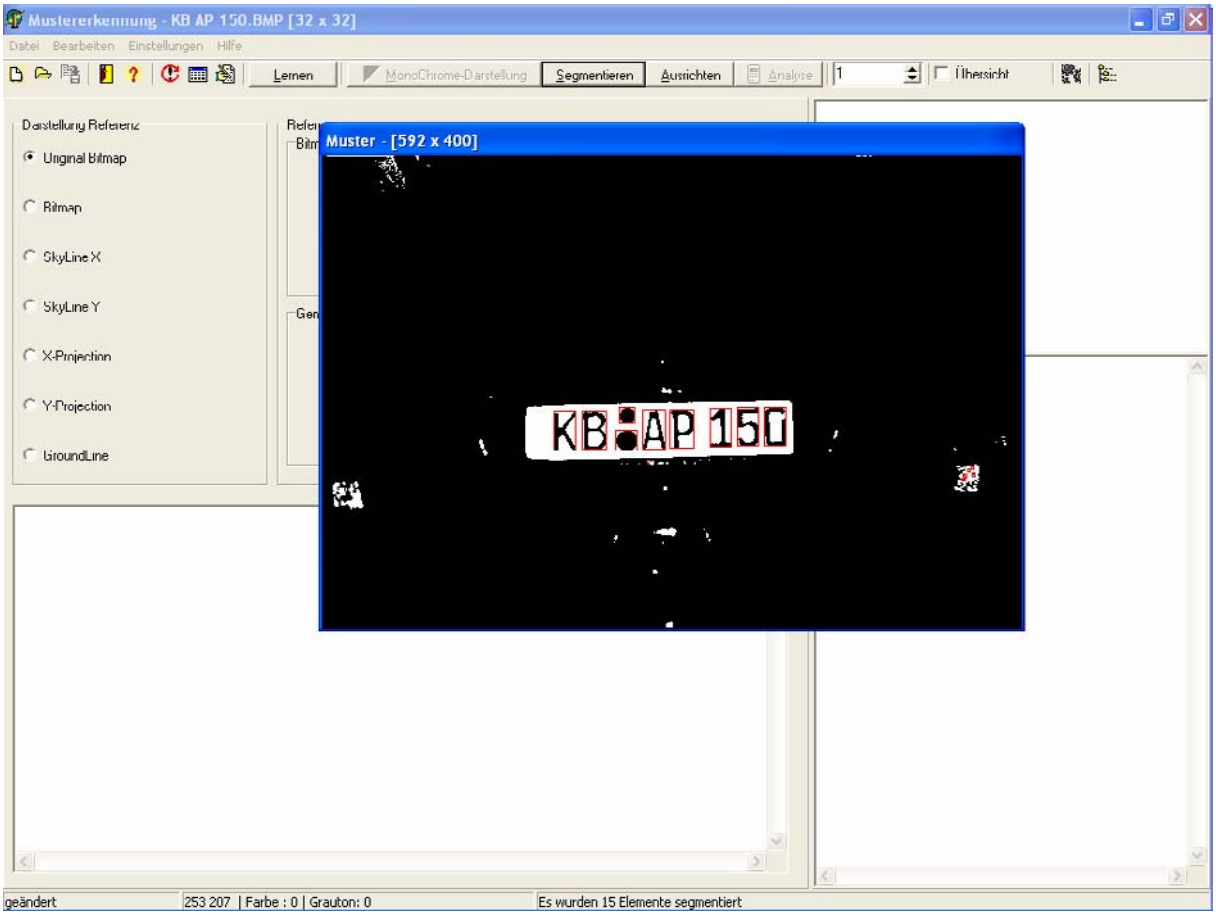


Abbildung 33 Segmentieren des Fotos

Wie man sieht, wurden 15 Segmente erkannt. Außer den „gewünschten“ Buchstaben/Ziffern allerdings auch der Zulassungs- und TÜV-Stempel, sowie einige kleinere Rechtecke insb. unten rechts.

Um die nicht gewünschten Segmente auszublenden, gibt es zunächst die Möglichkeit **nicht** das **gesamte** Foto zu segmentieren. Wenn z.B. aufgrund der Fahrzeugposition oder der Lage der Kamera die **ungefähre** Position des Autokennzeichens bekannt ist, so kann dieses Rechteck mit der Maus gezogen und über die rechte Maustaste als „Rechteck für Kennzeichen“ einstellt werden. Die Segmentierung beschränkt sich dann auf dieses markierte Rechteck.

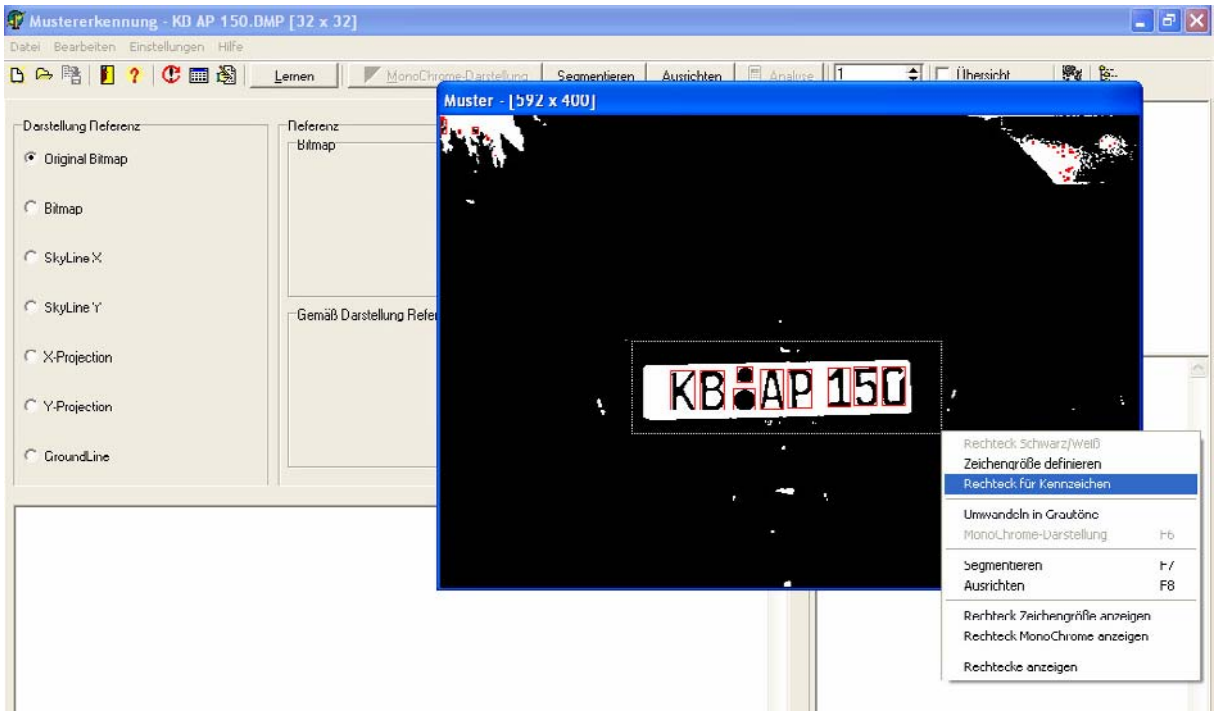


Abbildung 34 Einstellung des Rechtecks für das Kennzeichen

Weiterhin muss in den ***Einstellungen/Optionen*** diese Funktion aktiviert werden.

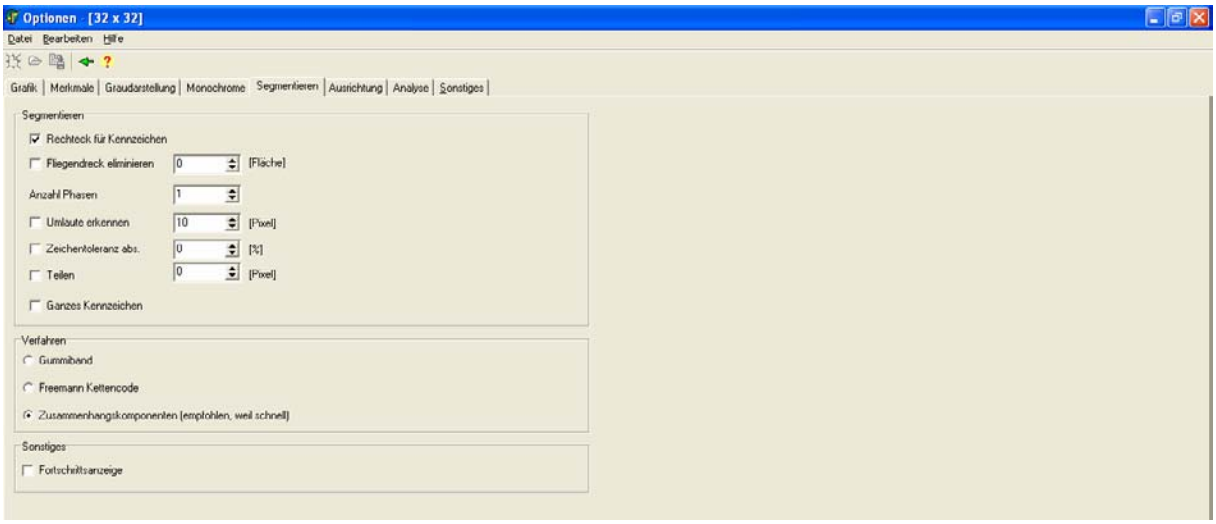


Abbildung 35 Aktivierung des Rechtecks für das Autokennzeichen

Nebenbei wird dadurch **auch** die Laufzeit des Segmentierungs-Algorithmus positiv beeinflusst. Welches Verfahren genutzt werden soll, kann hier ebenfalls eingestellt werden.

Ein erneutes Segmentieren findet jetzt nur noch Segmente im eingestellten Rechteck.

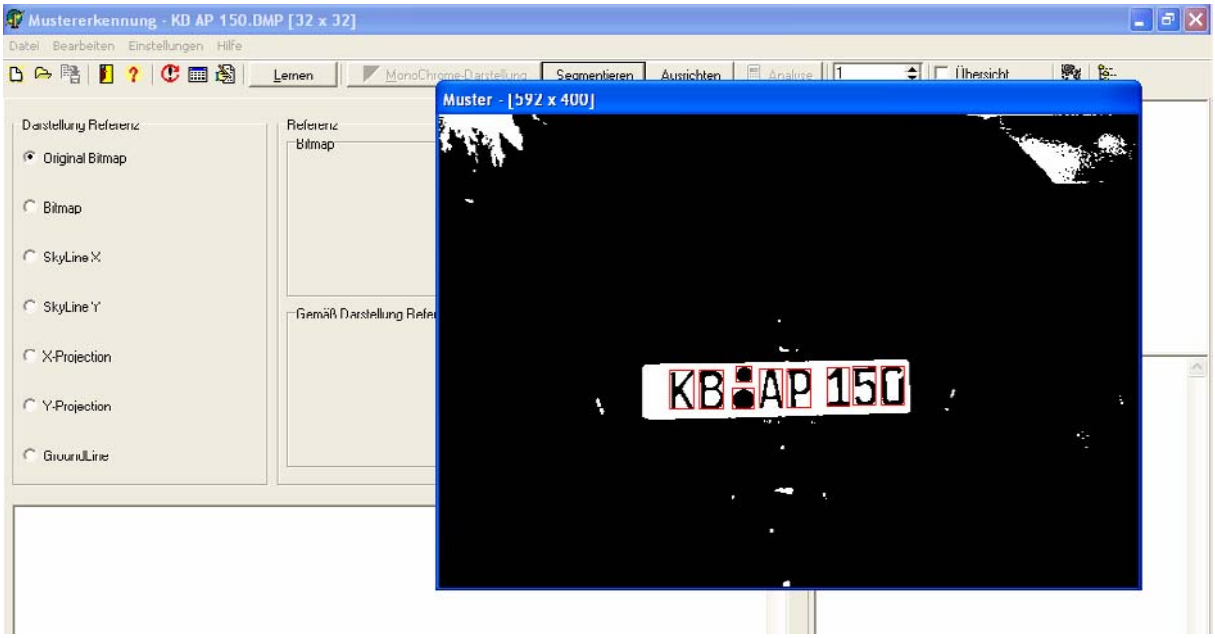


Abbildung 36 Segmentierung nach der Aktivierung des Kennzeichenrechtecks

Trotzdem können noch „unerwünschte“ Segmente gefunden werden. Häufig gibt es Verschmutzungen (z.B. durch Fliegendreck), die Befestigung des Kennzeichens durch Schrauben, etc. Die Fläche dieser Segmente ist meistens sehr klein im Verhältnis zu den „erwünschten“ Segmenten. In den /***Einstellungen/Optionen*** kann deshalb die Option „Fliegendreck“ aktiviert und eine Fläche (Breite x Höhe) in Bildpunkten angegeben werden. Gefundene Segmente mit einer Fläche kleiner gleich der Fliegendreck-Fläche werden dann ausgeblendet.

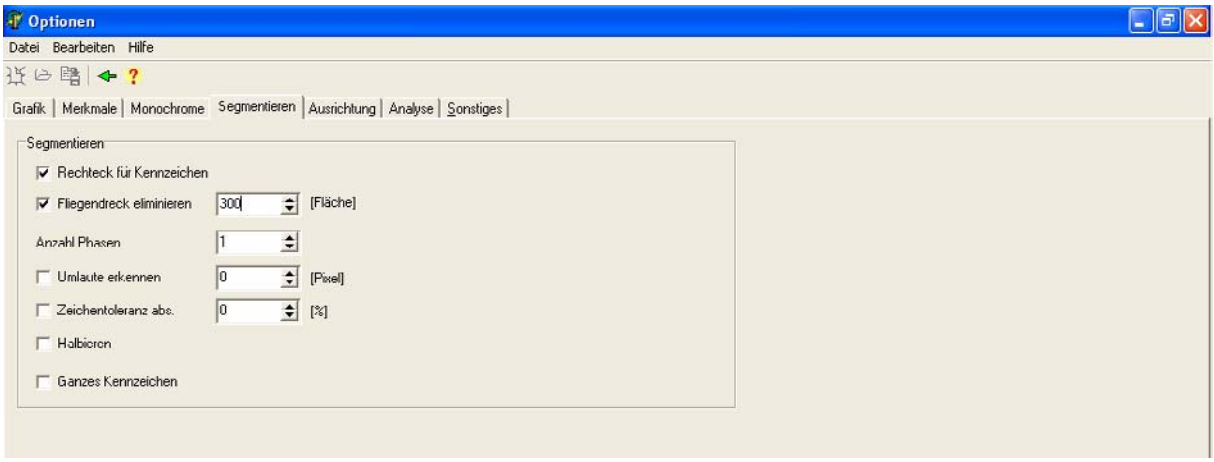


Abbildung 37 Aktivierung der Fliegendreckoption

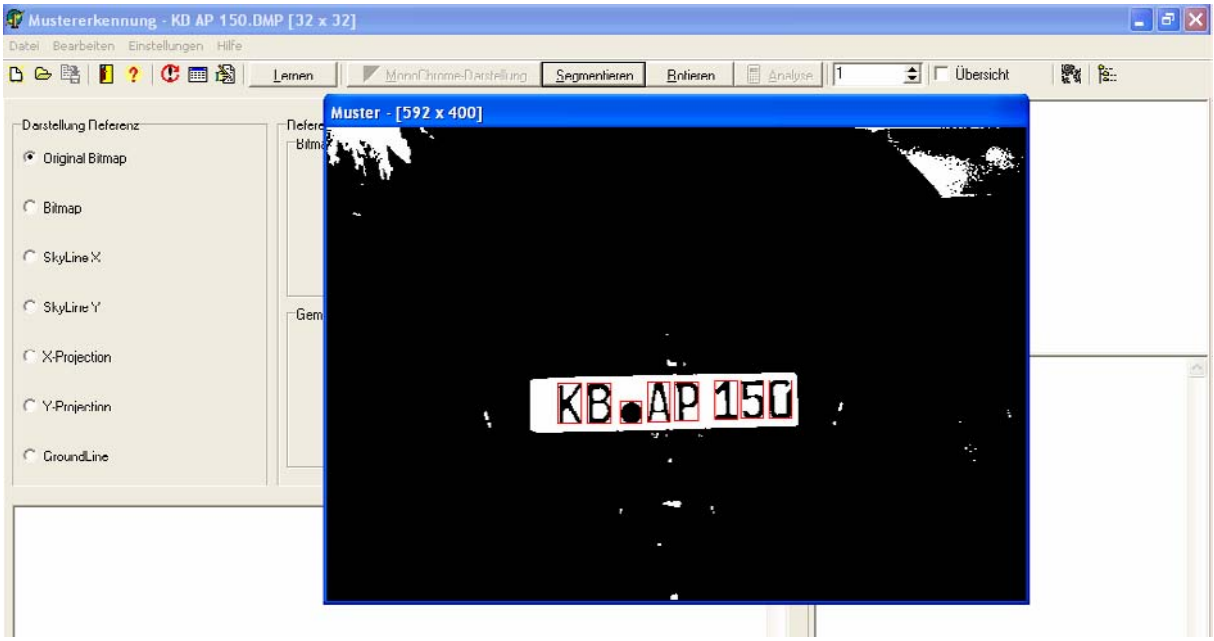


Abbildung 38 Erneutes Segmentieren mit der Option „Fliegendreck“

Wie man sieht, ist dabei sogar der TÜV-Stempel eliminiert worden.

Da die Buchstaben/Ziffern eines Autokennzeichens alle annähernd gleich groß sind kann man bei bekannter Fahrzeug- bzw. Kameraposition in den /***Einstellungen/Optionen*** die Zeichentoleranz aktivieren. Im Originalfoto kann man dann ein Rechteck mit der zu erwartenden Zeichengröße ziehen und dieses Rechteck als Zeichengröße definieren (rechte Maustaste). Segmente, die nicht dieser Zeichengröße entsprechen, werden dann bei der Segmentierung ausgeblendet.

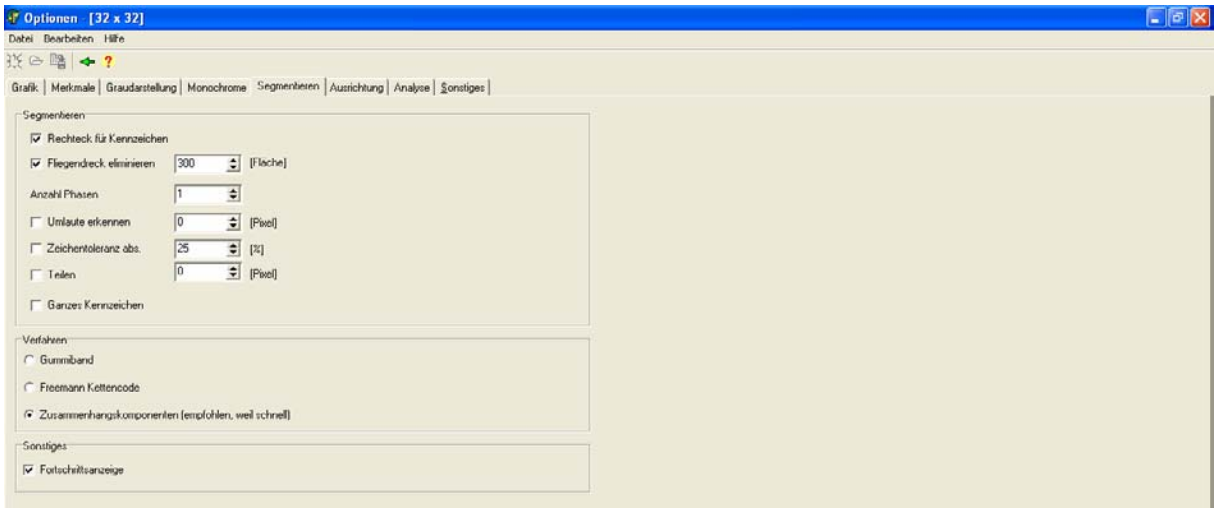


Abbildung 39 Einstellung der Segmentoptionen

Mit diesen Einstellungen werden alle gefunden Segmente mit weniger als 300 Punkten (Breite x Höhe) als „Fliegendreck“ ignoriert. Weiterhin dürfen die gefundenen Segmente um nicht mehr als ±25% von der eingestellten Zeichengröße (Breite, Höhe) abweichen.

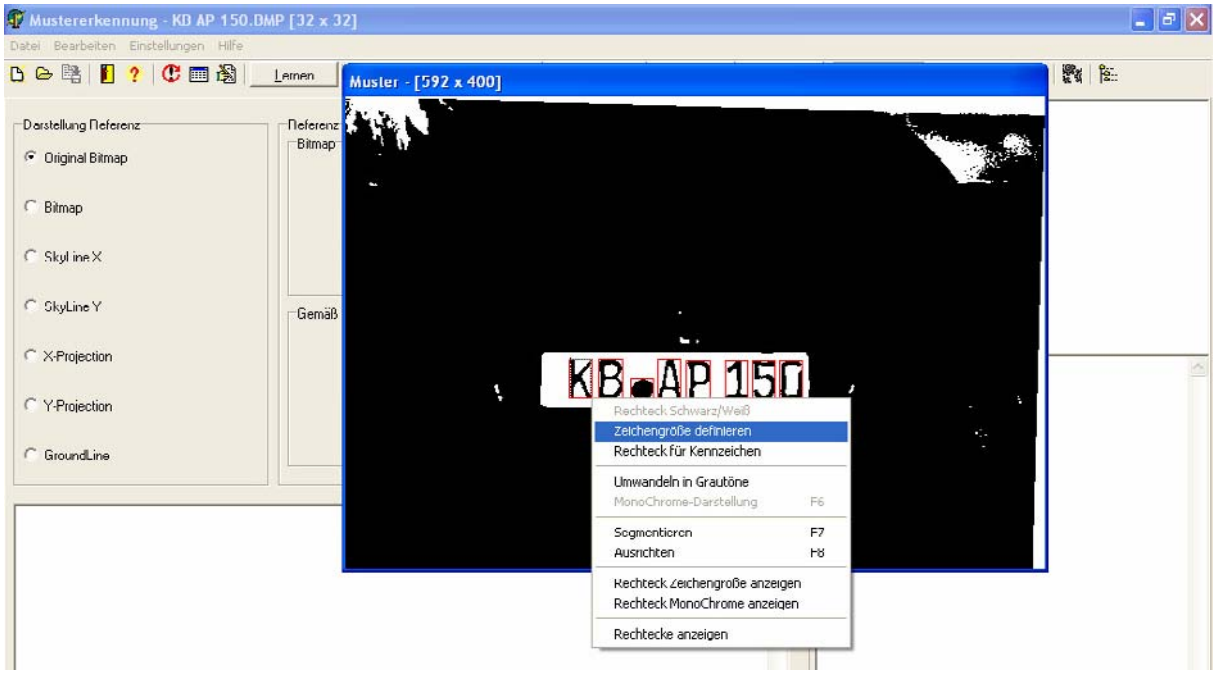


Abbildung 40 Einstellung der Zeichengröße

Hier wird um das K ein Rechteck mit der Maus gezogen und dieses Rechteck über die rechte Maustaste als Zeichengröße definiert.

Ein erneutes Segmentieren findet nun die gewünschten 7 Buchstaben/Zeichen. Der Zulassungstempel wurde nicht segmentiert, da er außerhalb der eingestellten Toleranzgrenze der Zeichengröße liegt.

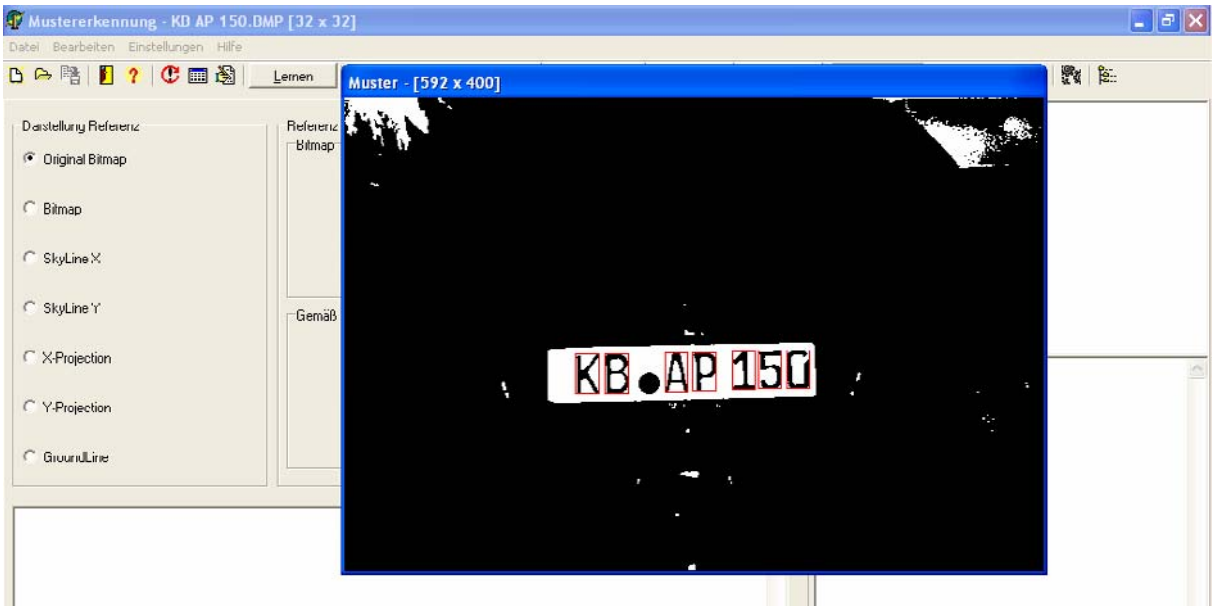


Abbildung 41 Die Segmentierung nach der Aktivierung der Zeichengröße

In den ***Einstellungen/Optionen*** gibt es noch weitere Filtereinstellungen für die Segmentierung (s. eigenes Kapitel).
Nun liegt das Foto in die Buchstaben/Ziffern des Autokennzeichens segmentiert vor.

Sollte bei der Aufnahme die Visierlinie der Kamera nicht rechtwinklig zum Kennzeichen gestanden haben, so kann das Foto ausgerichtet werden, um damit die Buchstaben/Ziffern rechtwinklig anzuordnen.
Dies geschieht über den Button **Ausrichten**. Dabei wird über die Koordinaten der linken unteren Ecke der Segmente eine lineare Regression durchgeführt. Über die ermittelte Steigung der Regressionsgeraden kann dann der Drehwinkel ermittelt werden, mit dessen **negativem** Wert das Foto gedreht wird.

Um nicht zu völlig unsinnigen Drehungen zu kommen, gibt es in den ***Einstellungen/Optionen*** einen Min- und Max-Winkel, in dessen Bereich der Drehwinkel liegen muss (sofern diese Option aktiviert wird).

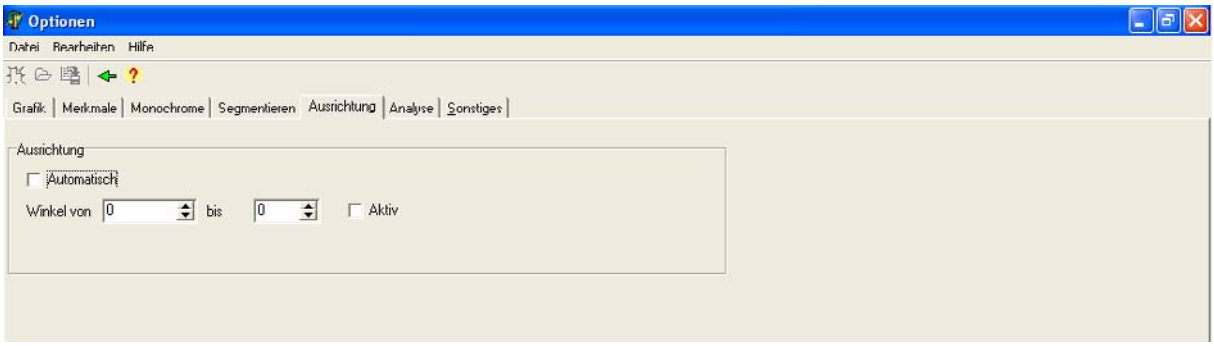


Abbildung 42 Einstellung der Ausrichtungsoptionen

Nun liegen die Segmente der Buchstaben/Ziffern in rechtwinkliger Form vor. Allerdings ist eine Texterkennung (Analyse) noch nicht möglich, da noch keine gelernten Referenzen von Buchstaben/Ziffern vorliegen, mit denen die Segmente verglichen werden könnten.

3.5 Lernen der Referenzen

Über den Button **Lernen** können nun die 7 gefundenen Segmente den gelernten Referenzen und der SpaCAM zugefügt werden. Auf dem Bildschirm kann komfortabel durch die gefundenen Segmente navigiert werden (Videotasten). Auf Wunsch kann das gewünschte Segment als Bitmap dem Referenzverzeichnis durch Angabe eines Dateinamen (*.bmp) zugefügt werden (***/Datei/Speichern unter ...***). Gleichzeitig wird dieses Segment in eine SpaCAM gelernt.

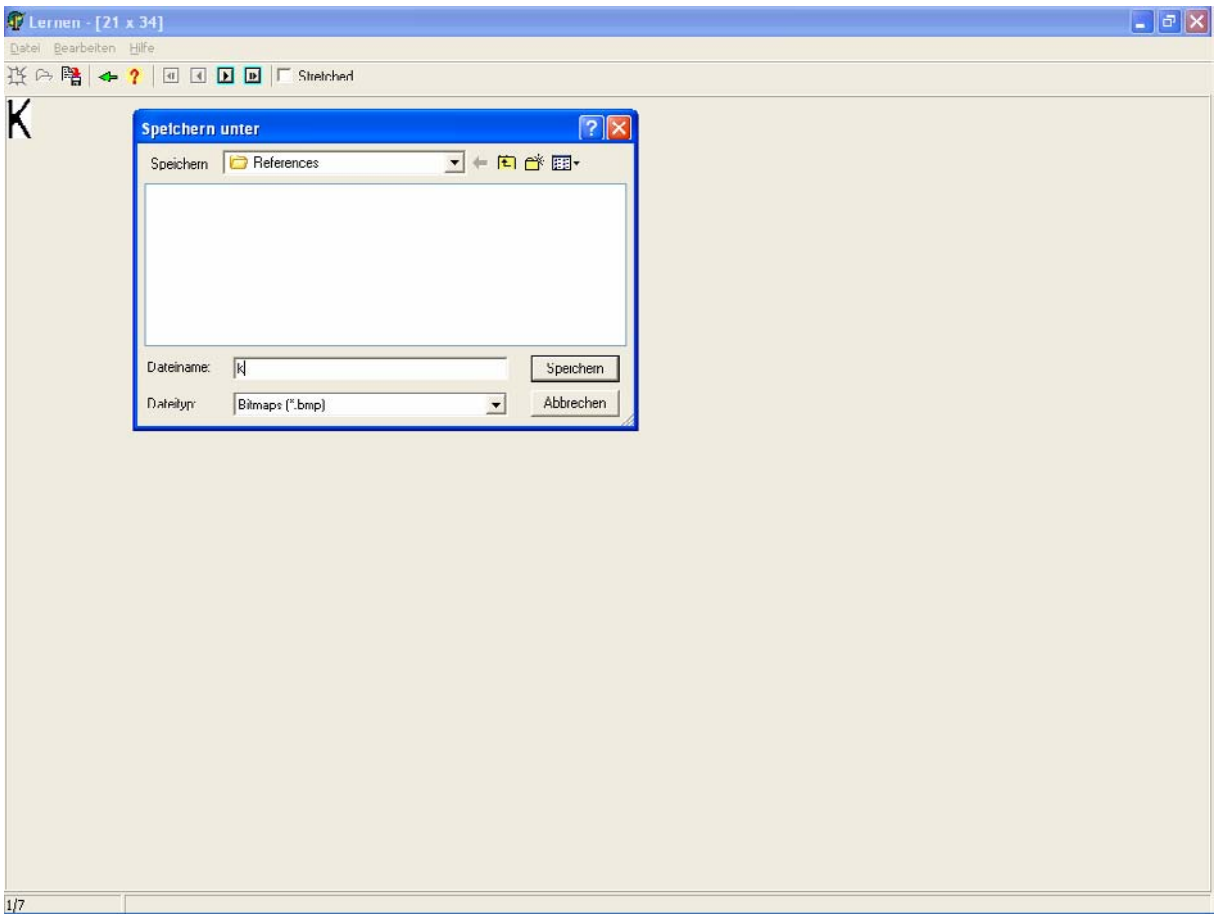


Abbildung 43 Lernen des Buchstabens K

So können alle Zeichen des Autokennzeichens gelernt werden. Sie werden anschl. im Referenzverzeichnis sichtbar.

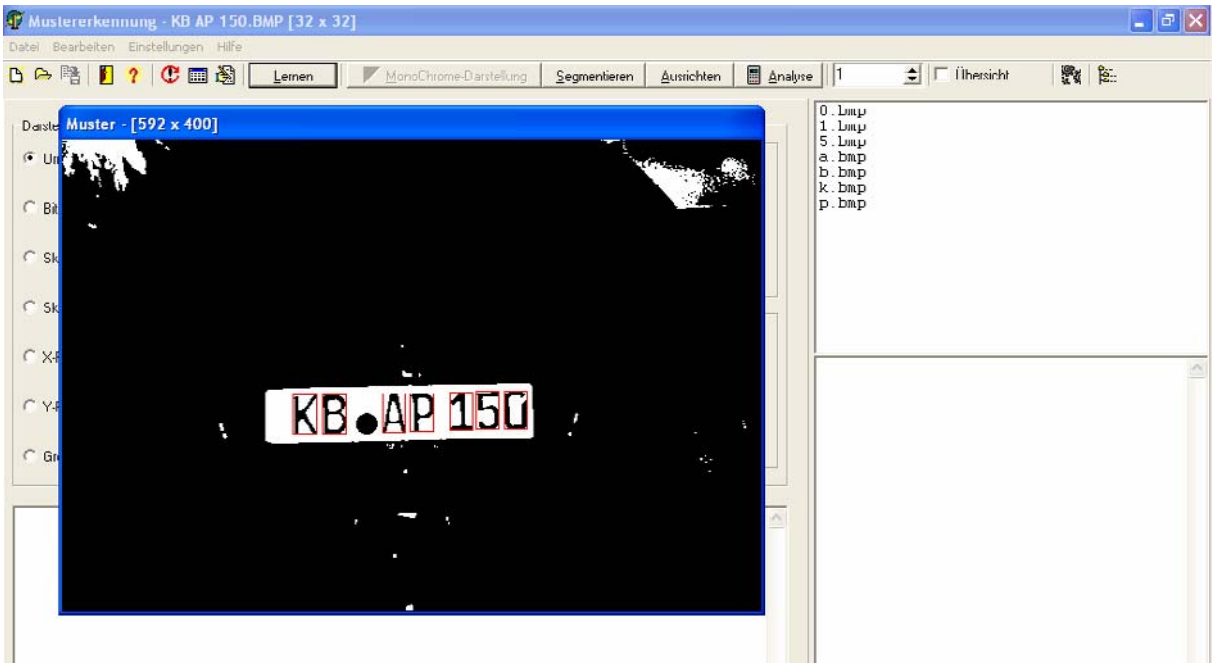


Abbildung 44 Alle Zeichen sind gelernt

3.6 Texterkennung (Analyse)

Über den Button **Analyse** kann nun versucht werden, das Autokennzeichen zu analysieren.

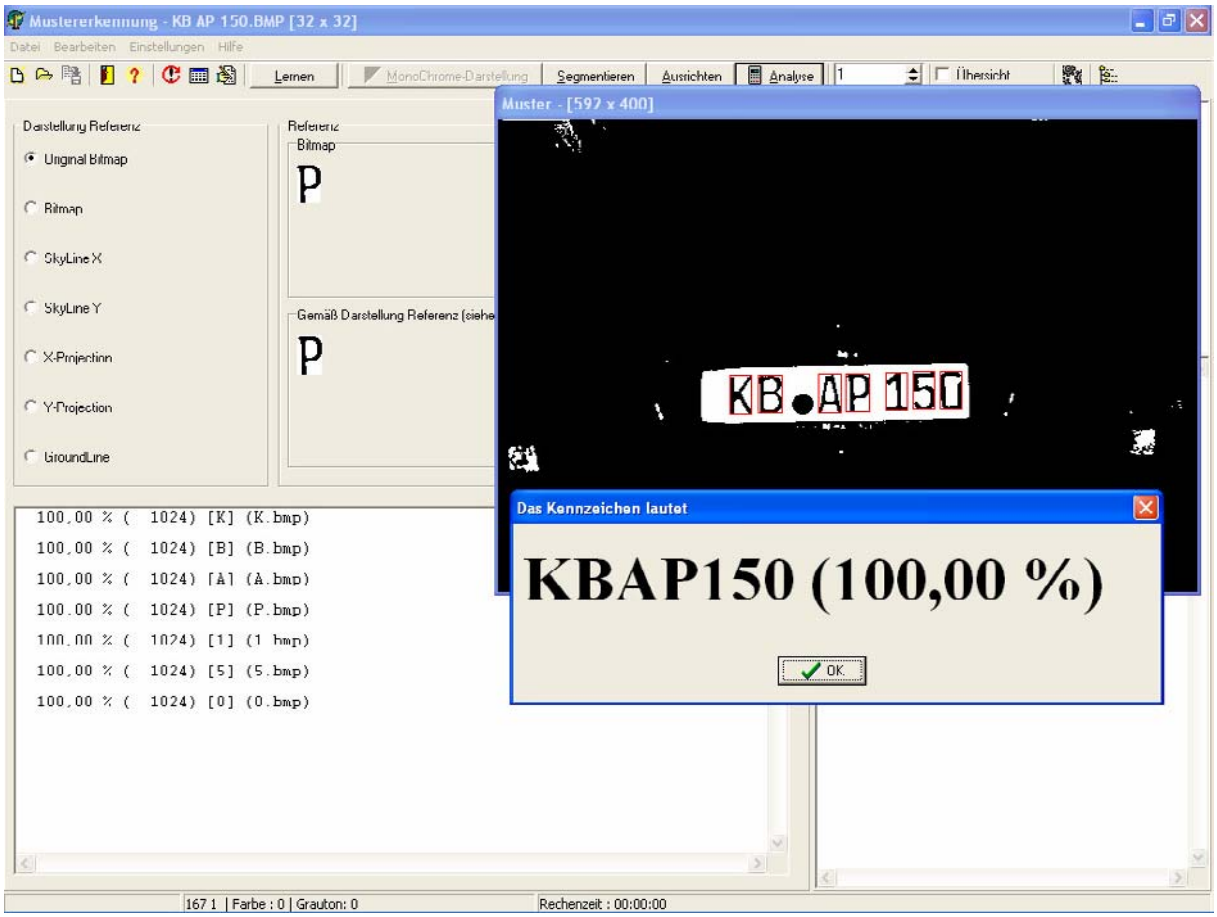


Abbildung 45 Analyse des Autokennzeichens

Dabei wird jedes Segment mit den gelernten Referenzen verglichen und die Übereinstimmung in % ermittelt. Die Referenz mit der besten Übereinstimmung wird ausgewählt. Da bei diesem Beispiel alle Referenzen aus den gefundenen Segmenten gelernt wurden, wird natürlich jedes Segment mit 100% Übereinstimmung gefunden und damit das Autokennzeichen zu 100% erkannt.

Nun kann diese bescheidene Referenzmatrix auf andere Kennzeichen untersucht werden. Dazu lädt man das Foto eines weiteren Autokennzeichens.

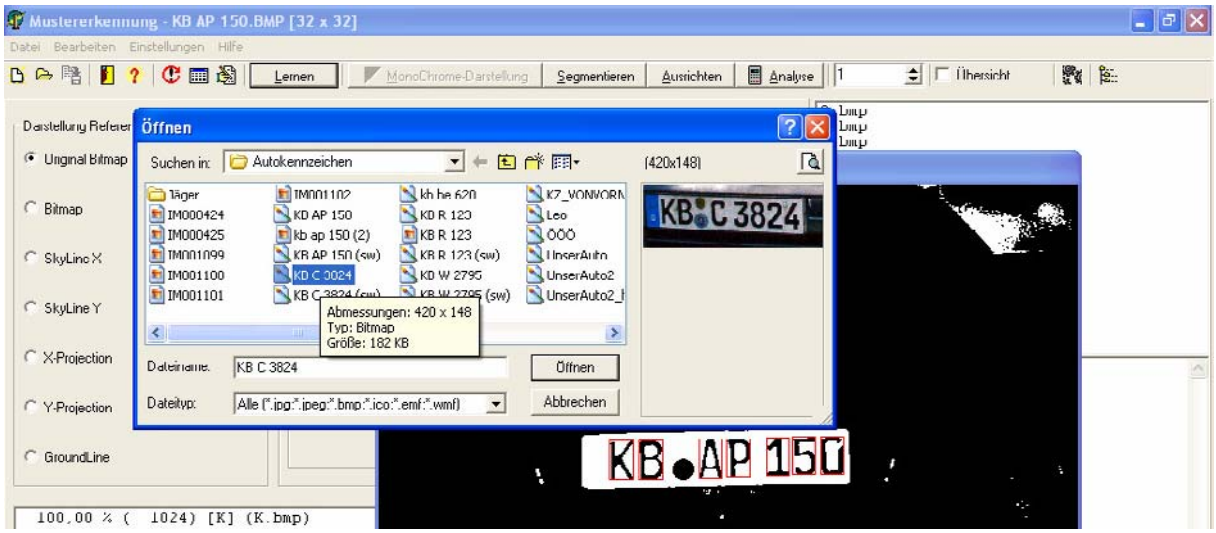


Abbildung 46 Laden eines weiteren Fotos

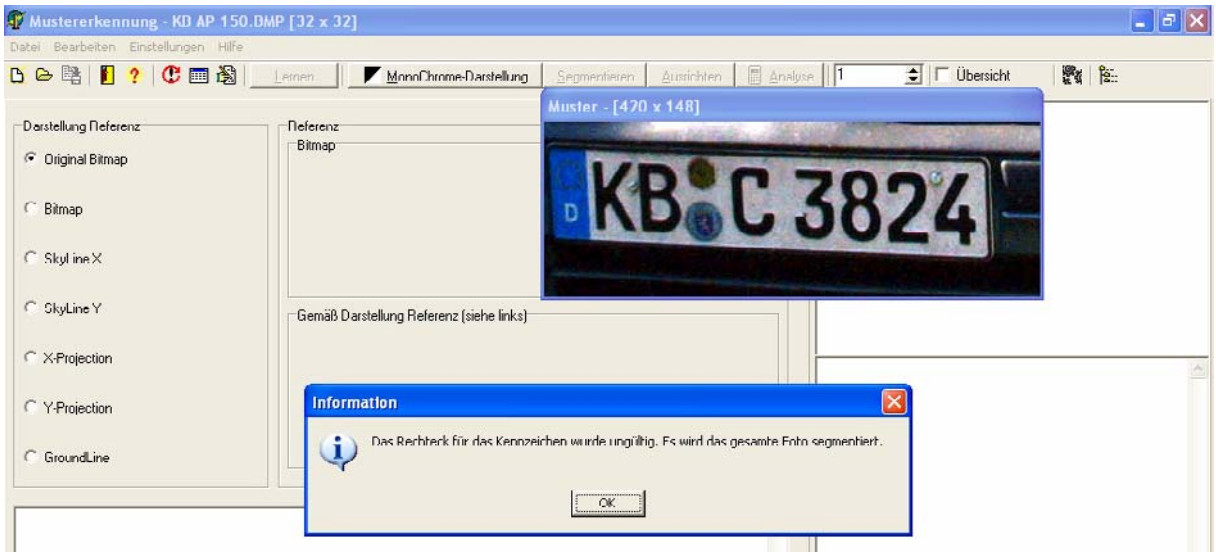


Abbildung 47 Einige Parameter sind ungültig geworden

Da dieses Foto wesentlich kleiner ist als das vorige, ist das Rechteck für das Kennzeichen ungültig geworden. Es muss neu definiert oder die Option ausgeschaltet werden. Möglicherweise ist auch die Zeichengröße nicht mehr korrekt, da dieses Foto aus wesentlich geringerer Entfernung aufgenommen wurde. Dann muss die Zeichengröße neu definiert werden.

Die Segmentierung ergibt:

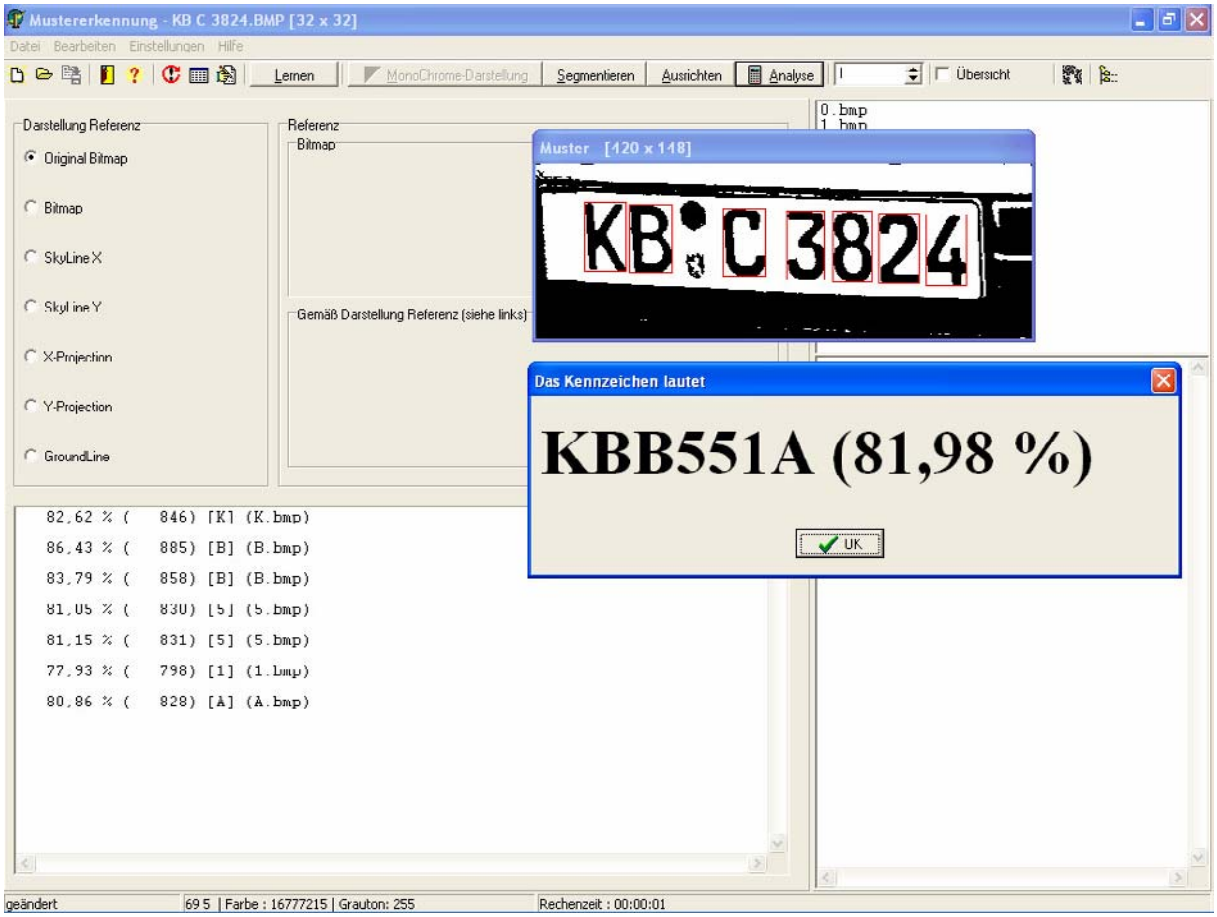


Abbildung 48 Die Segmentierung des neuen Kennzeichens

Wie man sieht, werden K und B mit einer Übereinstimmung von über 80% erkannt. C, 3, 8, 2 und 4 wurden bisher nicht gelernt, trotzdem versucht das System die „beste“ Übereinstimmung zu finden; allerdings mit doch recht geringen Prozentsätzen.
Somit könnten die Referenzen aus diesem Kennzeichen um die fehlenden Zeichen C, 3, 8, 2 und 4 nach o.g. Muster ergänzt werden. Da ein Autokennzeichen aus lediglich 29 Buchstaben (A..Z und die Umlaute) sowie den 10 Ziffern bestehen kann, dürften recht wenige Musterkennzeichen genügen, um alle möglichen Buchstaben/Ziffern Referenzen zu lernen.

Möchte man wissen, wie gut welches Segment mit welcher Referenz übereinstimmt, so genügt ein Doppelklick auf das gewünschte Segment (z.B. K).

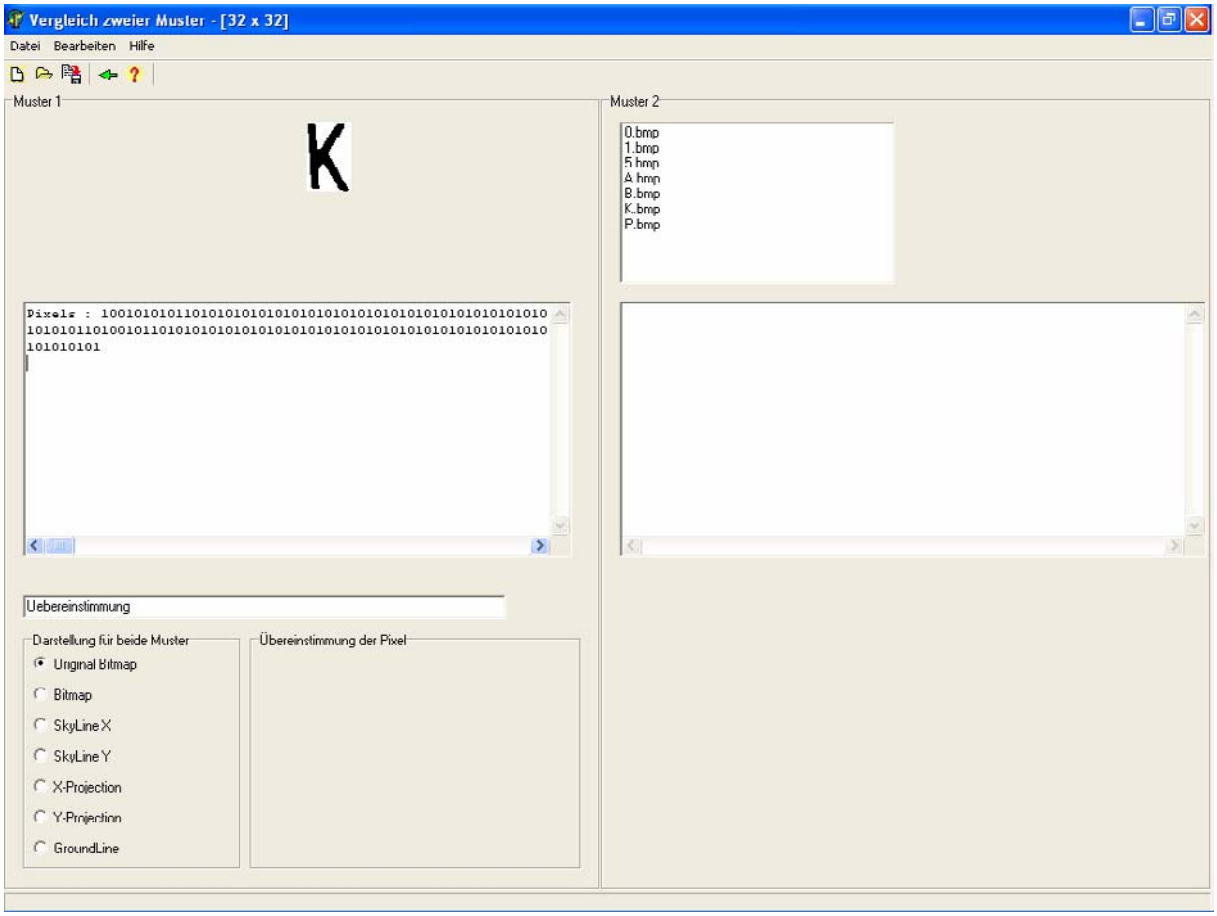


Abbildung 49 Doppelklick auf „K“ zur weiteren Analyse

Im linken Teil wird das angeklickte Segment dargestellt. Im rechten Teil sieht man eine Liste der gelernten Referenzen. Klickt man nun z.B. im rechten Teil das K an, so kann man die Übereinstimmung analysieren. In der Mitte werden für beide Muster der SpaCAM-Vektor angezeigt. Unten links kann auch das gewünschte Merkmal angewählt werden. Im Rahmen „Übereinstimmung der Pixel“ werden beide Muster XOR-verknüpft übereinander gelegt; d.h. dort wo beide Pixel identisch sind (also schwarz-schwarz oder weiß-weiß) ist das übereinander gelegte Muster schwarz sonst weiß. Der Schwarzanteil entspricht damit der Übereinstimmung der Pixel.

In beiden bisher betrachten Fällen, wurde je Segment nur die Referenz angezeigt, die die beste Übereinstimmung hat (1 Treffer). Interessant kann aber auch sein, die zweit-, dritt-, ... beste Referenz zu sehen.

Dazu gibt es in der oberen Leiste eine Einstellung.

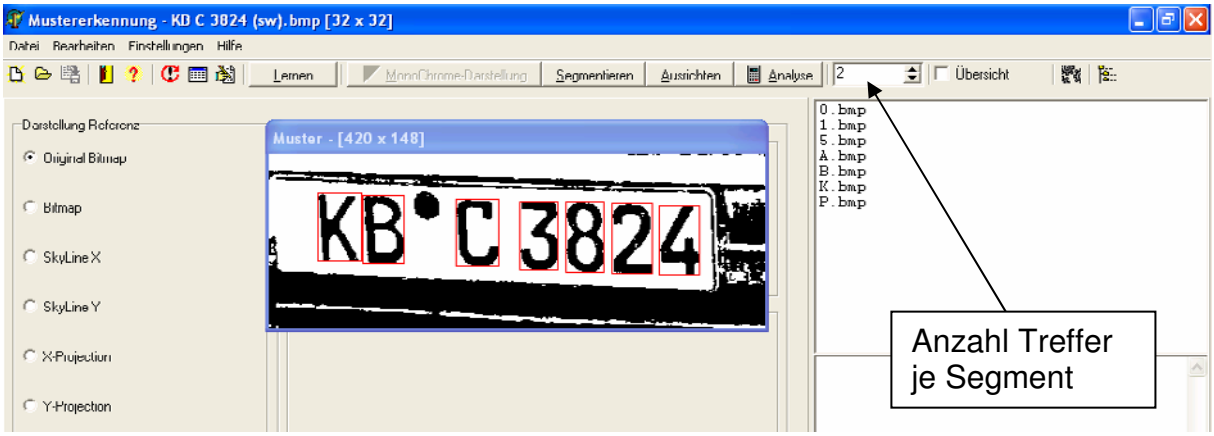


Abbildung 50 Erhöhung der Trefferzahl je Segment auf 2

Die Analyse ergibt:

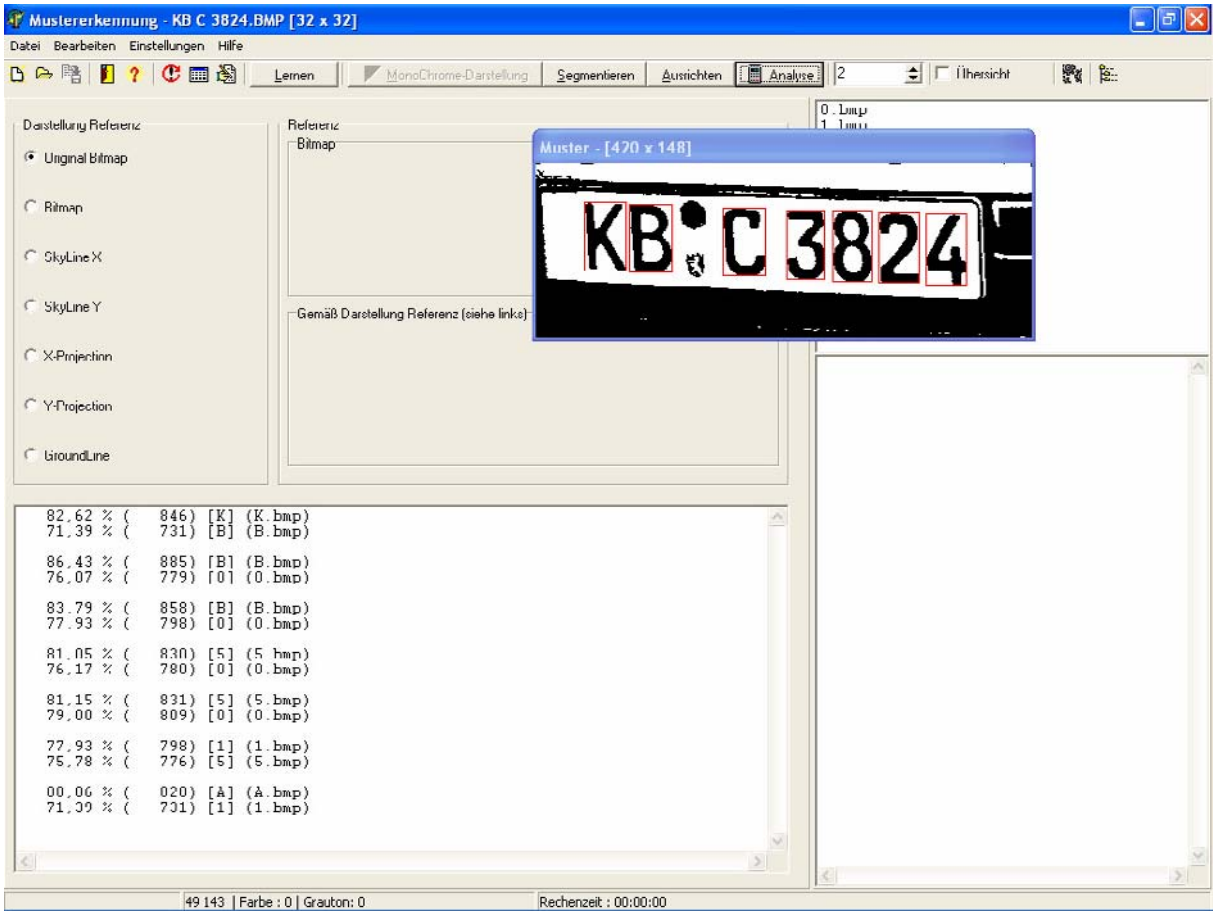


Abbildung 51 Die Analyse mit 2 Treffern je Segment

Somit wird für das K als zweitbester Treffer B erkannt. Für das B als zweiter Wert die 0, usw.

Mit diesen Informationen können nun rekursiv alle möglichen Kombinationen von Autokennzeichen mit den Güten ermittelt werden. In diesem Fall gibt es bei 7 Segmenten und 2 Treffern 2^7 = 128 mögliche Autokennzeichen. Zur Ansicht kann man in der oberen Leiste **Überblick** ankreuzen und die Analyse erneut starten. Da hierfür u.U. recht lange Laufzeiten zu erwarten sind, wird abgefragt, ob diese Analyse wirklich gewünscht wird.

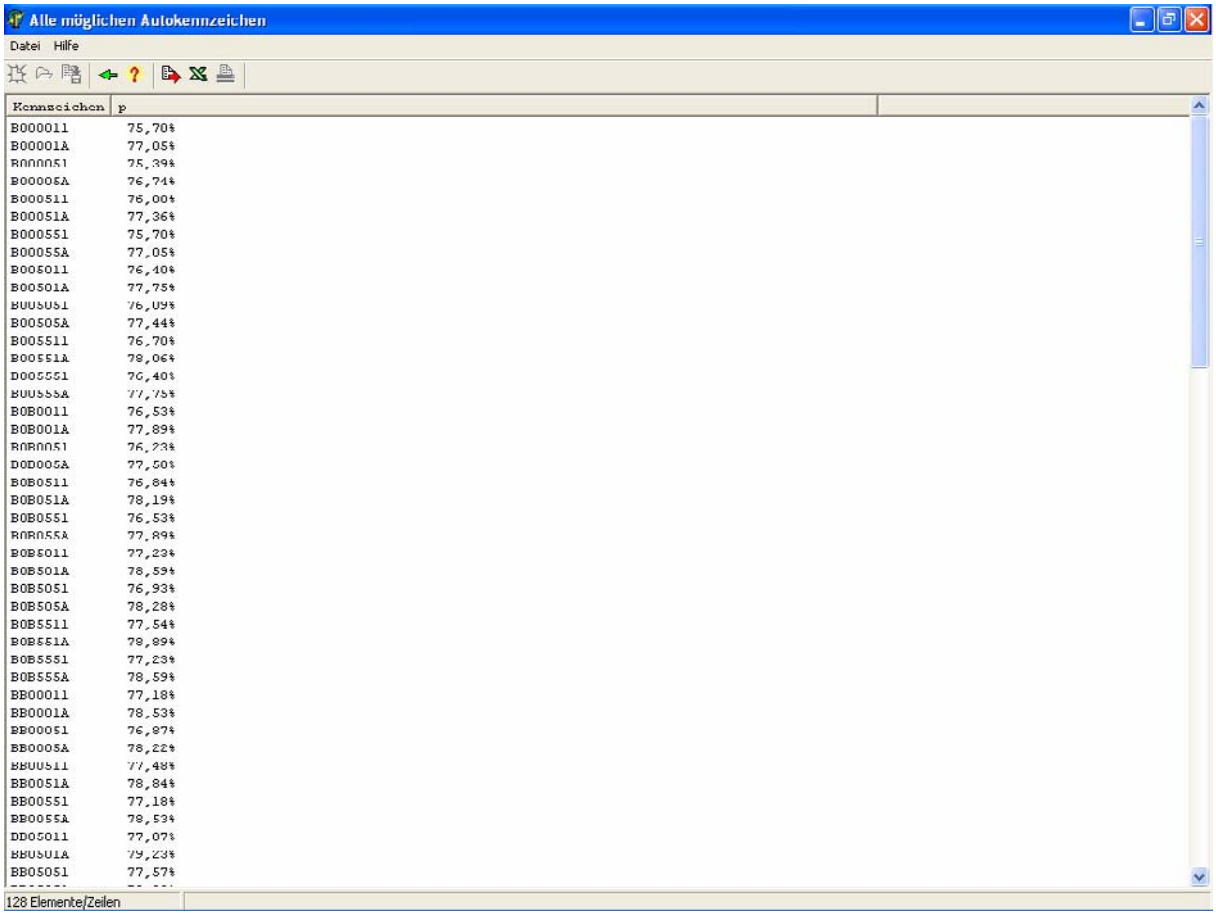


Abbildung 52 Die 128 Kombinationen möglicher Autokennzeichen

In der ersten Spalte steht das mögliche Kennzeichen, in der zweiten Spalte die Güte für dieses Kennzeichen, als **arithmetisches** Mittel der Güten der Einzelsegmente. Wie von ähnlichen Darstellungen unter WINDOWS gewohnt, kann durch Klick auf die Spaltenüberschrift für Analysezwecke nach der Spalte auf- bzw. absteigend sortiert werden. Auch besteht die Möglichkeit, diese Tabelle nach EXCEL zu exportieren und so auch externen Programmen zur weiteren Bearbeitung zur Verfügung zu stellen.

3.7 Weitere Analysewerkzeuge

3.7.1 Analyse einer Referenz

Wenn ein Foto geladen ist, sollte dies zunächst über ***/Datei/Neu*** entfernt werden. Klickt man nun z.B. auf die Referenz 0.bmp so wird im oberen Mittel des Bildschirms das Bitmap der gelernten 0 dargestellt. Im linken oberen Teil können nun verschiedene Merkmale der gelernten Referenz dargestellt werden.

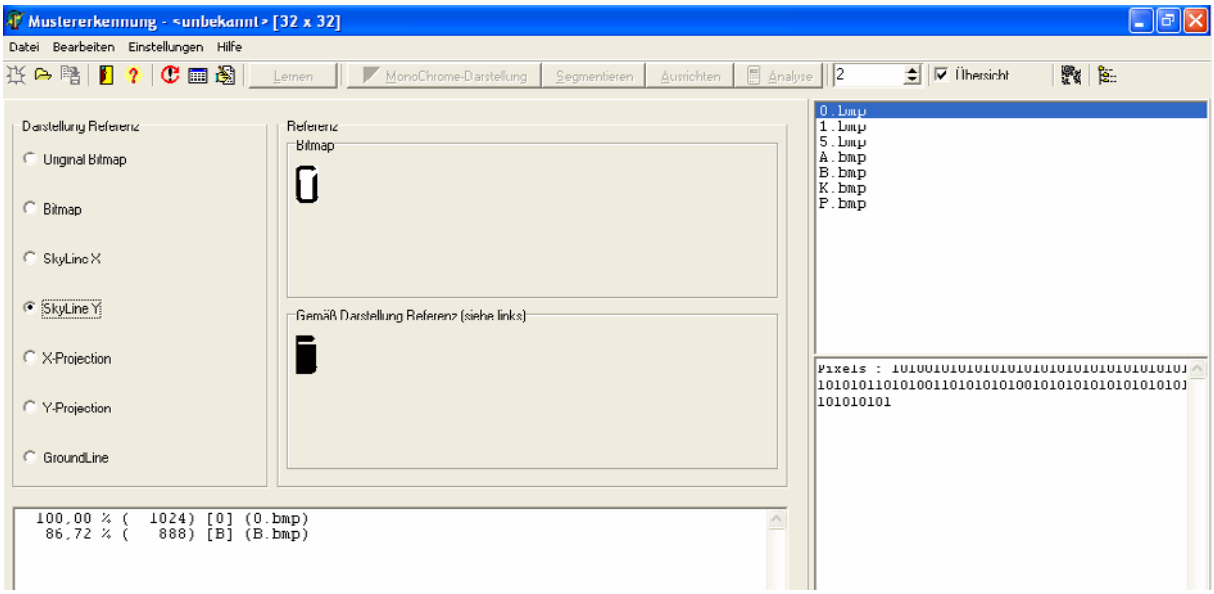


Abbildung 53 Die SkyLineY der „0“

Im unteren Teil wird die Übereinstimmung mit sich selbst und dann (je nach eingestellter Trefferanzahl) mit der nächst schlechteren Referenz dargestellt. Rechts werden die Merkmalsausprägungen lt. eingestellten Merkmalen (***Einstellungen/Optionen***) dargestellt. Da hier „nur“ der Pixelvergleich eingestellt wurde, wird auch nur dieses Merkmal angezeigt. Letztlich ist diese Angabe (binär und spärlich codiert) die entsprechende Information in der zugrunde liegenden SpaCAM.

3.7.2 Vergleich zweier beliebiger Referenzen

Über */**Bearbeiten/Vergleich*** zweier Referenzen können gezielt **zwei Referenzen** miteinander verglichen werden.

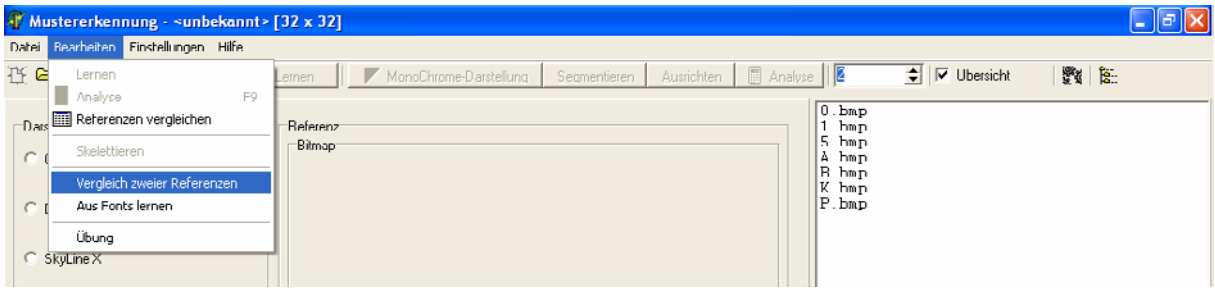


Abbildung 54 Auswahl des Referenzenvergleichs

Dabei öffnet sich ein mittig geteiltes Fenster. In jedem Fenster kann eine der Referenzen ausgewählt werden. Für die beiden so ausgewählten Referenzen werden die unterschiedlichen Merkmale und deren Übereinstimmung angezeigt (Übereinstimmung ist ein schwarzer Pixel). Diese Maske entspricht der o.g. Nur ist jetzt im linken Teil ebenfalls ein gelerntes Muster aus der Referenz sichtbar und nicht das aus einer Segmentierung.

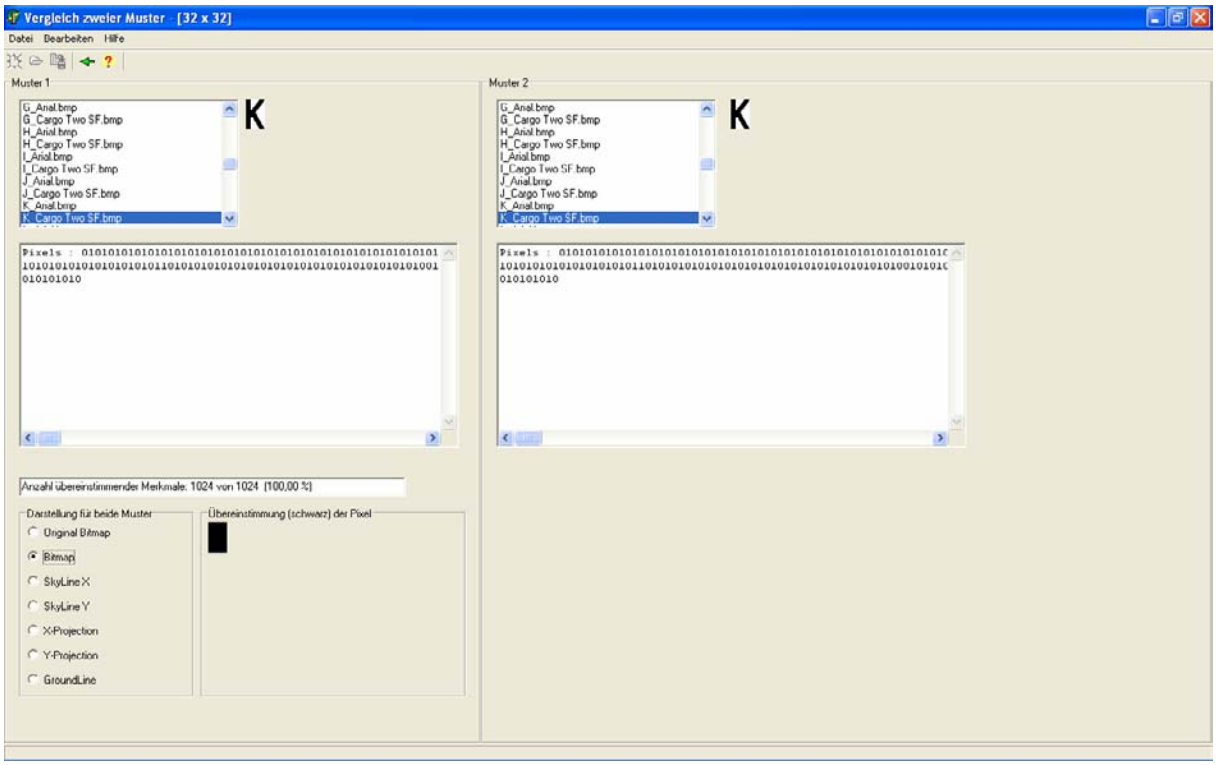


Abbildung 55 Vergleich zweier Referenzen

Da hier K mit K verglichen wird ist die Übereinstimmung der Pixel ein schwarzes Rechteck.

3.7.3 Referenzenvergleich

Um zu sehen, wie unterschiedlich **alle** Referenzen untereinander sind, bzw. wie gut sie sich gegenseitig diskriminieren, kann man über ***/Bearbeiten/Referenzen vergleichen*** eine entsprechende Matrix generieren.

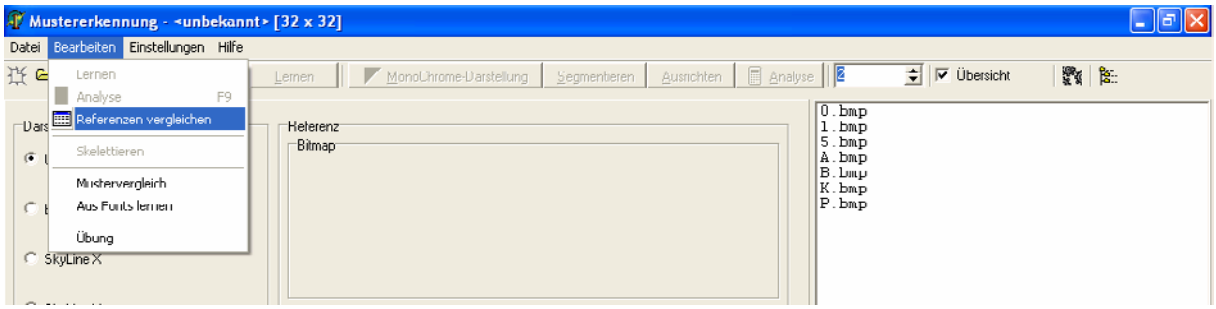


Abbildung 56 Vergleich der Referenzen

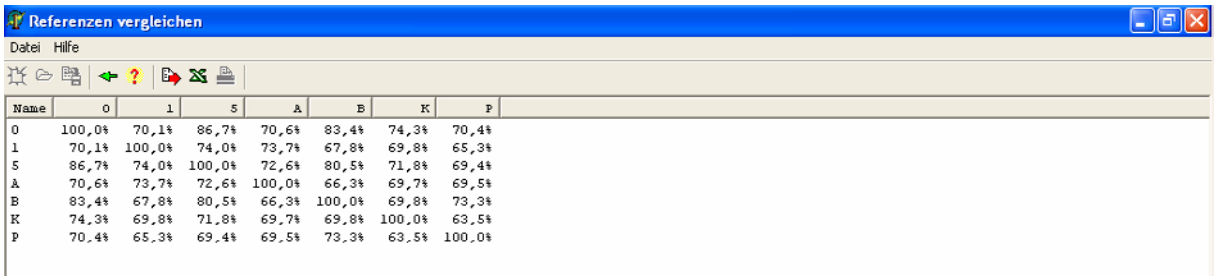


Abbildung 57 Ergebnis des Referenzenvergleichs

3.8 Besonderheiten von Autokennzeichen

Das System kann nur dann Muster erkennen, wenn zuvor Referenzen gelernt wurden. Dies kann wie oben gezeigt geschehen. Dabei ist der Grad der Erkennung bzw. Übereinstimmung abhängig von der Güte der Referenz. Somit sollte in den Aufbau der Referenzdatenbank größte Sorgfalt gelegt werden. Dies kann dadurch erfolgen, dass besonders „saubere“ Fahrzeuge unter optimalen Lichtverhältnissen fotografiert werden. Da es sich bei den Autokennzeichen um lediglich 39 unterschiedliche Zeichen handelt, könnte man auch zu einer Zulassungsstelle gehen und sich ein (oder auch mehrere) Autokennzeichen mit allen Buchstaben/Ziffern besorgen und diese dann unter optimalen Bedingungen fotografieren. Es gibt aber den oben beschriebenen, wesentlich eleganteren Weg, die Referenzen aus Fonts zu lernen.

Ruft man diese Routinen für alle gewünschten Buchstaben/Ziffern auf, wird eine optimale Referenzmatrix gelernt. Dieses Verfahren eignet sich somit nicht nur für **Autokennzeichen**, sondern für alle über Fonts druckbare Zeichen. Im Programm gibt es dazu den Menüpunkt ***/Bearbeiten/Aus Fonts lernen.***

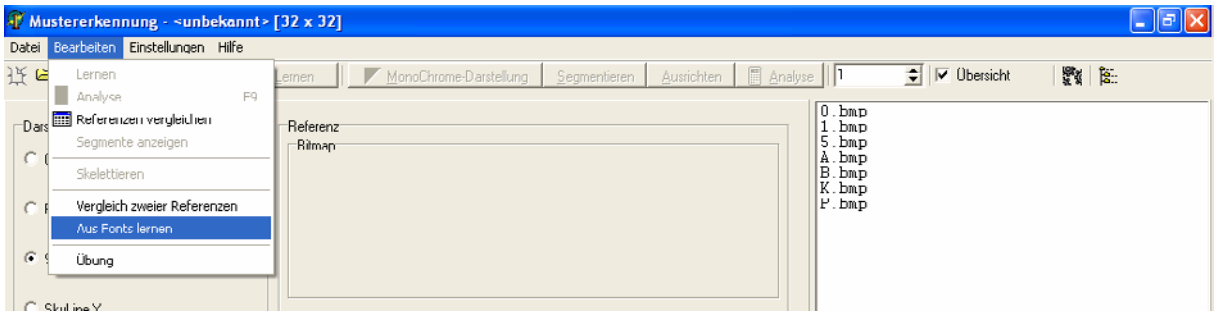


Abbildung 58 Anwahl der Funktion zum Lernen aus Fonts

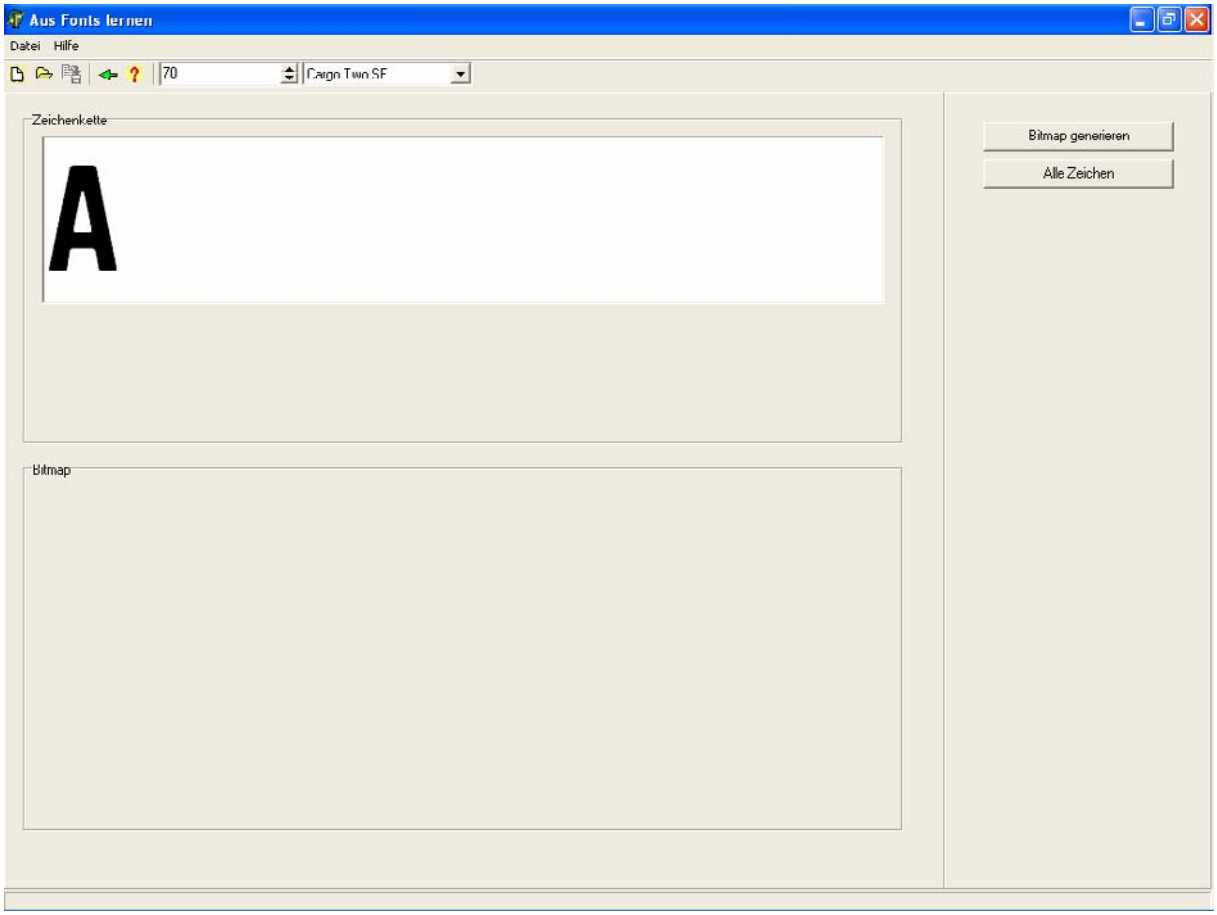


Abbildung 59 Bildschirmmaske zum Lernen aus Fonts

Wird nun z.B. als Zeichenkette der Buchstabe A eingetragen, als Zeichengröße 70 vorgegeben und der Button „Bitmap generieren“ gedrückt, so erscheint der Buchstabe A in der Default-Schriftart in der Mitte des Bildschirms.

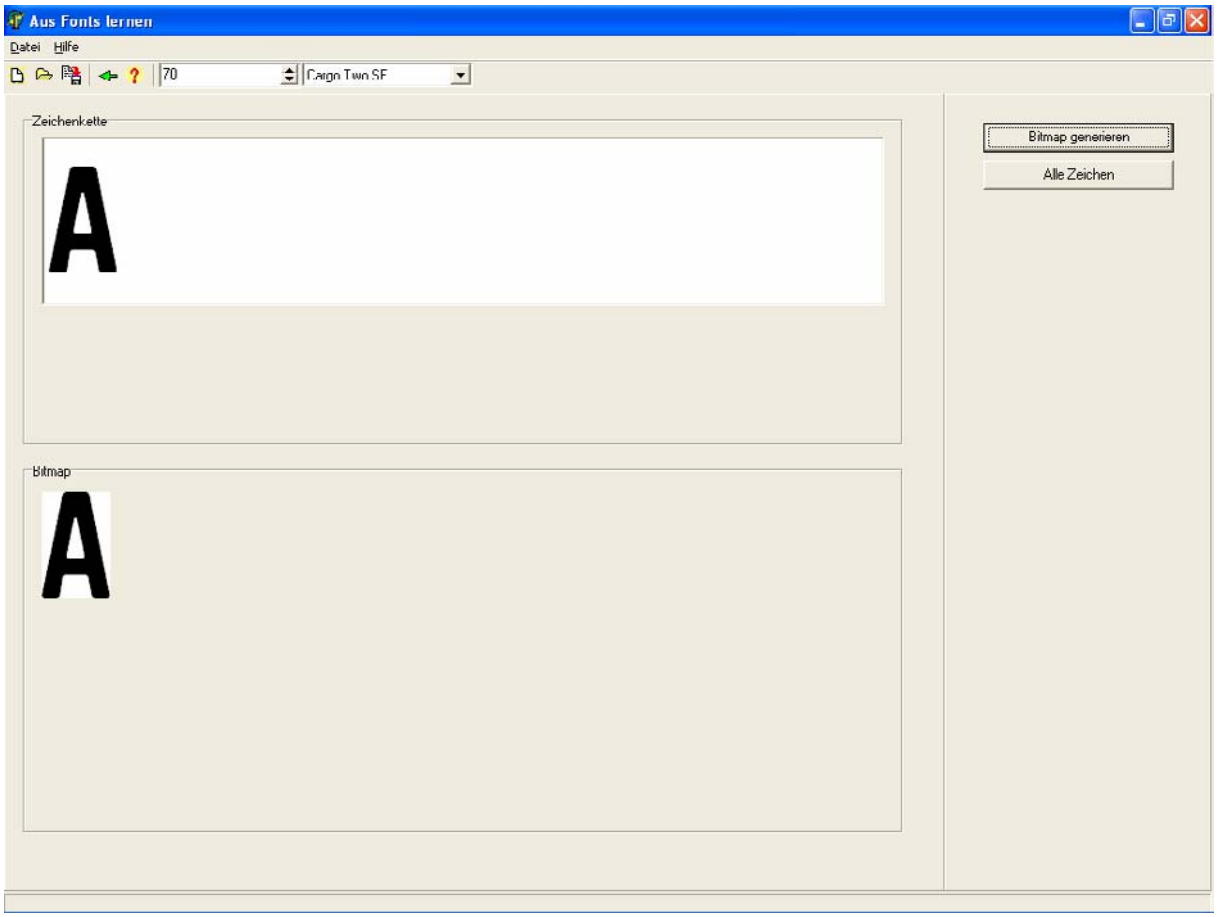


Abbildung 60 Das generierte Bitmap aus „A“ in der Schriftart für Autokennzeichen

Dieses Muster könnte jetzt über ***/Datei/Speichern*** z.B. als A.bmp gespeichert und damit den Referenzen zugefügt werden. Gleichzeitig würde dann A in die SpaCAM gelernt. Da bereits eine Datei A.bmp aus dem Lernen von KB-AP 150 existiert, würde hiermit allerdings das bisherige A überschrieben. Das kann aber sinnvoll sein, da das neue A völlig frei von „Fehlern“ ist.

Über den Button **„Alle Zeichen“** werden die Buchstaben A-Z, die Ziffern 0-9 und die Umlaute Ä,Ö und Ü automatisch gespeichert und damit den Referenzen und der SpaCAM zugefügt.
Verlässt man dieses Bild, so wurden die Referenzen um die gelernten Muster aus der Schriftart ergänzt. Zur Unterscheidung wird hinter dem Buchstaben der Name der

Schrift getrennt durch ein „_“ (das sog. Versionskennzeichen) als Dateinamen eingetragen. Damit lassen sich auch unterschiedliche Schriftarten gleichzeitig in **einer** SpaCAM verwalten.

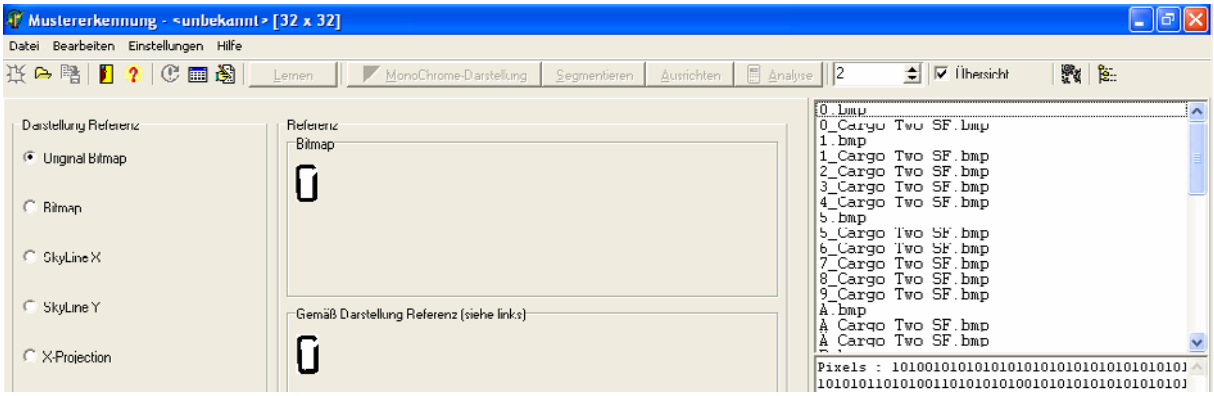


Abbildung 61 Referenzen wurden erweitert

Sollen die „alten“ Referenzen entfernt werden, so genügt ein Klick und über die rechte Maustaste ein „Löschen“. Gleichzeitig wird dieses Muster natürlich auch aus der SpaCAM entfernt.

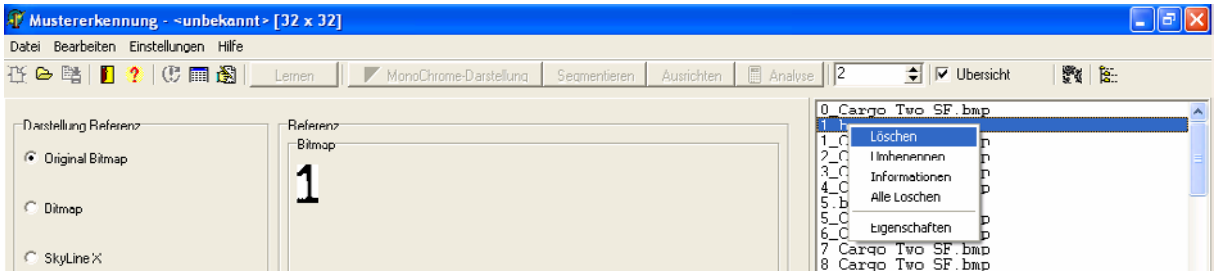


Abbildung 62 Löschen der „alten“ Referenz der „1“

Nun kann ein anderes Foto geladen werden.

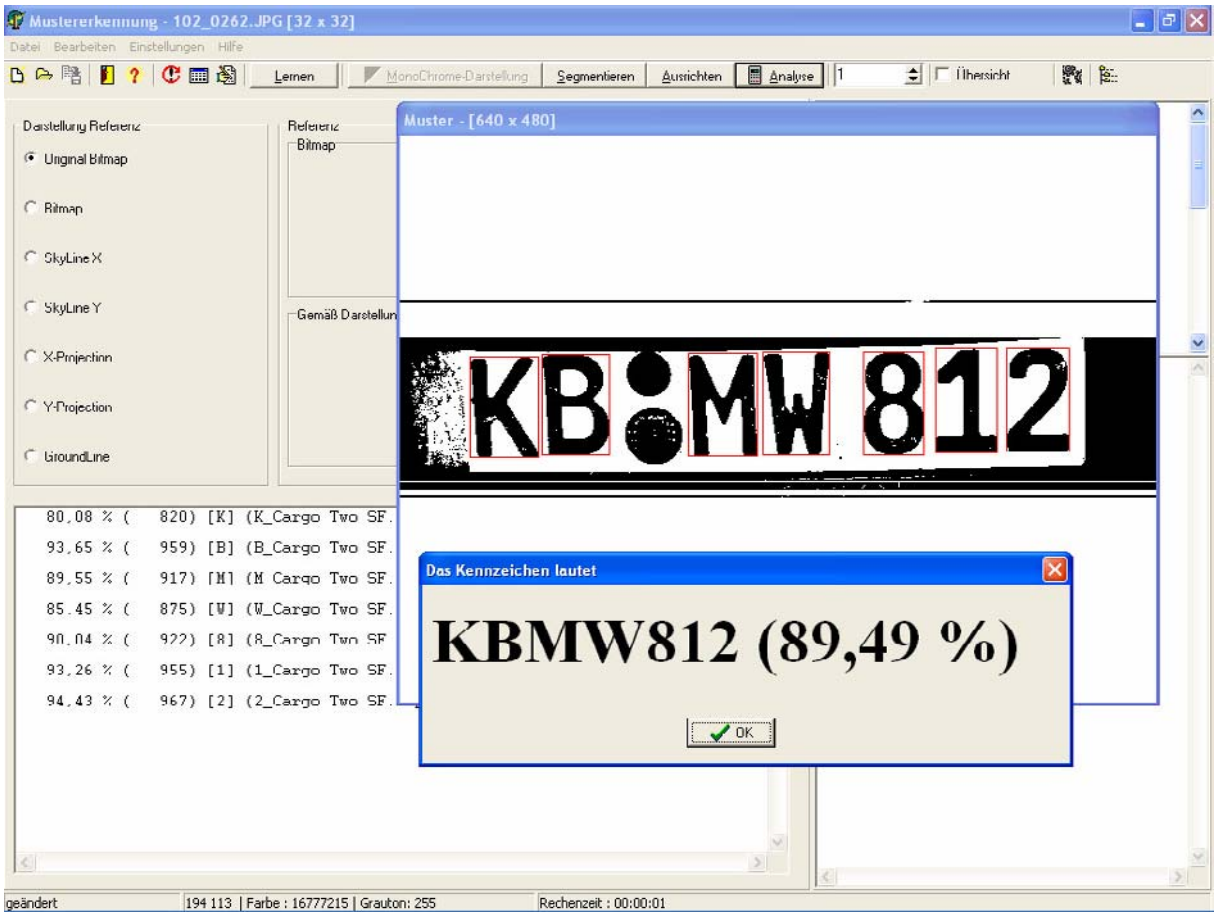


Abbildung 63 Laden eines anderen Fotos und Analyse

Da nun alle Referenzen über die Schriftart gelernt wurden, werden jetzt alle Zeichen des Autokennzeichens erkannt.

3.9 Optionen

3.9.1 Einleitung

Über */**Einstellungen/Optionen*** kann das System in vielen Bereichen so angepasst werden, dass auch bei ungünstigen Bedingungen optimale Ergebnisse erreicht werden können. Der Bildschirm enthält eine Tabelle mit 8 Reitern, deren Inhalte im Folgenden beschrieben werden soll.

3.9.2 Graphik

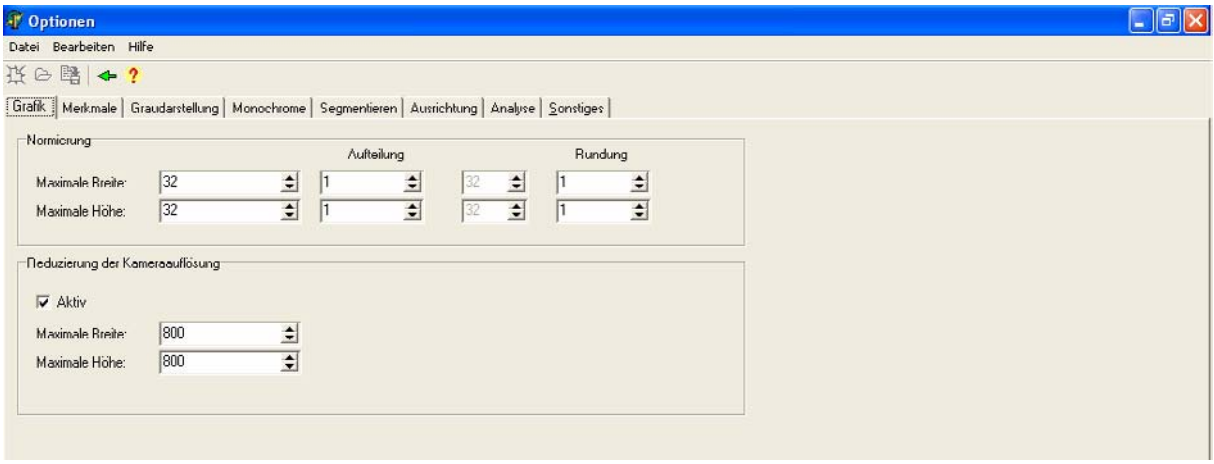


Abbildung 64 Einstellung der Graphikoptionen

Hier kann eingestellt werden, auf welche Abmessungen die gelernten bzw. segmentierten Zeichen normiert werden sollen, damit sie untereinander vergleichbar werden. Dabei kann die Breite und die Höhe eingestellt werden. Weiterhin kann angegeben werden, ob das Bild weiter unterteilt werden soll. In der Praxis hat es sich als sinnvoll herausgestellt, die Werte der Merkmale zu runden, um leichte Unschärfen auszublenden und letztlich einen Beitrag zur Fehlertoleranz zu leisten.
Die mittlerweile sehr hohen Auflösungen von Digitalkameras (> 2 Mio. Pixels) sind für diese Applikation nicht nötig und erzeugen ggf. unnötigen Rechenaufwand. Somit kann bei gewünschter Aktivierung die Reduktion der Auflösung auf die gewünschte Größe eingestellt werden.

In o.g. Einstellung werden die Muster auf eine 32 x 32 Bitmap normiert. Die Merkmale sollen nicht unterteilt werden. Die Rundung erfolgt auf 1. Weiterhin sollen die Originalbilder auf eine Auflösung von max. 800 x 800 Pixel (unter Beachtung des Originalverhältnisses von Höhe und Breite) reduziert werden.

3.9.3 Merkmale

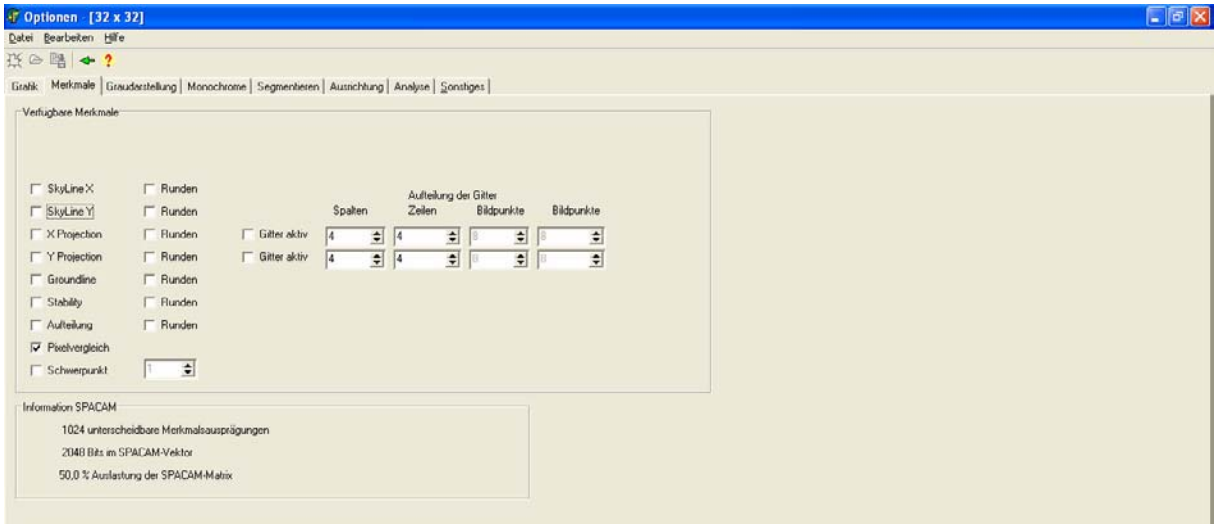


Abbildung 65 Einstellungen der Merkmalsoptionen

Hier können die gewünschten Merkmale ausgewählt und ggf. die Rundung angegeben werden. Gleichzeitig wird die Größe der SpaCAM-Vektoren bzw. –Matrix ausgewiesen.

In dieser Einstellung soll lediglich das Merkmal des Pixelvergleichs genutzt werden. Bei einer Normierung auf 32 x 32 Bits ergeben sich dabei 1024 unterscheidbare Merkmalsausprägungen. Der SpaCAM-Vektor hat dann 2048 Bits.

3.9.4 Graudarstellung

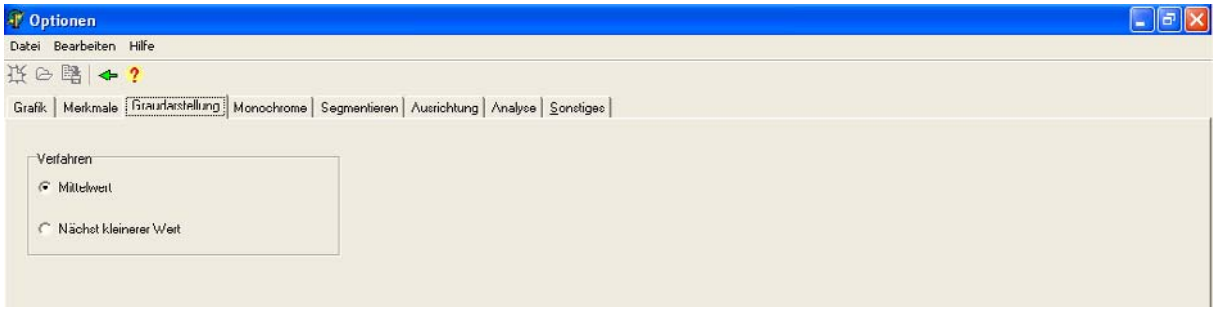


Abbildung 66 Einstellung für die Grauwertumwandlung

Hier kann eingestellt werden, ob der Grauwert über den Mittelwert der 3 Intensitäten R,G,B oder über den nächst kleineren Wert, bei dem alle Intensitäten gleich sind, ermittelt wird.

3.9.5 Monochrom

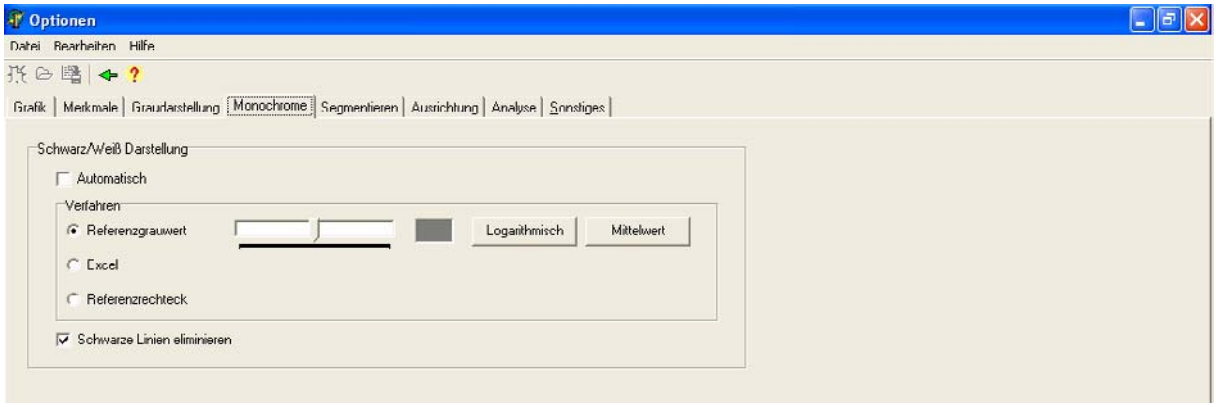


Abbildung 67 Einstellungen der Monochromoption

Die Originalbilder werden i.d.R. als Farbfotos vorliegen. Zur Segmentierung müssen diese monochrom (über den Umweg von i.d.R. 256 Grauwerten) dargestellt werden. Wird „automatisch“ eingestellt, so wird dies sofort beim Laden eines Fotos erledigt.
Zur Umwandlung eines Farbfotos nach Monochrom stehen 3 Verfahren zur Verfügung:

1. Es wird ein Schwellwert definiert. Alle Pixel, die dunkler als dieser Referenzwert sind, werden schwarz, die anderen werden weiß dargestellt. Der gewünschte Referenzgrauwert kann über den Schieberegler eingestellt werden. Weiterhin stehen zwei Buttons für voreingestellte Werte zur Verfügung:
    - Logarithmisch: Der Schwellwert beträgt $(\log_2 1.5)*255 = 149$
    - Mittelwert : Der Schwellwert beträgt 256/2 = 128
2. EXCEL bietet die Möglichkeit, Bilder nach Monochrom umzuwandeln. Wird diese Option angewählt, wird das Foto per „OLE“ über die Zwischenablage nach EXCEL übertragen, dort umgewandelt und zurück geladen.
3. Es kann im Originalbild ein Rechteck definiert werden, dessen gewogenes Mittel der Grautöne den Schwellwert darstellt. Dies kann z.B. das kennzeichenumschließende Rechteck sein.

Zur Laufzeitoptimierung kann es sinnvoll sein, vollständig schwarze Linien (waagerecht und senkrecht) im Originalfoto zu eliminieren.

3.9.6 Segmentieren

Abbildung 68 Einstellungen zur Segmentoption

Standardmäßig wird versucht, das gesamte Originalfoto zu segmentieren. Wenn aber bekannt ist (z.B. aufgrund von standardisierten Parkplätzen der Fahrzeuge und/oder einem festen Platz der Kamera), in welchem rechteckigem Bereich das Kennzeichen zu finden ist, kann dies hier eingestellt werden. Im Originalbild kann dann dieses Rechteck definiert werden.

Wie bereits beschrieben, können „Fliegendreck“, Schrauben etc. dadurch entfernt werden, dass Segmente, die unter der eingestellten Fläche für den „Fliegendreck“ bleiben, bei der Bearbeitung auf weiß gestellt werden. Bei guten Fotos genügt häufig ein Durchlauf zur Segmentierung. Soll aber z.B. der Fliegendreck entfernt werden, so können 2 oder auch mehr Durchläufe nötig werden. Dies kann als Phase eingestellt werden.

Autokennzeichen können Umlaute enthalten. Das System erkennt natürlich bei der Segmentierung die Punkte über den (ebenfalls segmentierten) Vokalen A,O und U, sofern die Toleranz des Fliegendrecks nicht zu großzügig eingestellt wurde. Das Programm versucht nun, die Punkte mit den zugehörigen Vokalen in Verbindung zu bringen, in dem beide Rechtecke (Punkt, Vokal) übereinander gestellt werden. Wenn die Gesamthöhe beider Rechtecke der Höhe eines „normalen“ Zeichens entspricht, so wird daraus ein Umlaut. Die Toleranz bei der Höhenaddition kann hier definiert werden.

Das monochrome Bild eines Originalfotos enthält auch nach der Bereinigung von Fliegendreck i.d.R. eine Fülle von segmentierten Rechtecken. Ist der Standort des Fahrzeugs und/oder der Kamera annähernd konstant, so ist aber die Größe eines Zeichens (Buchstabe/Ziffer) annähernd bekannt und kann im Originalbild eingestellt werden. Um genügend fehlertolerant zu sein, kann zusätzlich eine Toleranz in % angegeben werden; d.h. die gefundenen Segmente werden nur dann weiter betrachtet, wenn sich deren Breite und Höhe in der Sollgröße ± der eingestellten Toleranz in % bewegt.

Trotz aller Filter kann es passieren, dass die Zeichen eines Autokennzeichens nicht einzeln segmentiert werden. Häufig werden dann zwei oder mehr Zeichen nebeneinander als ein Segment erkannt. Der Zwischenraum zwischen den Zeichen enthält häufig nur wenige Bildpunkte je Spalte. Liegt die Anzahl dieser Bildpunkte unter der hier eingegebenen Anzahl Pixel, so wird an diesen Stellen das Segment geteilt.

Sollten Referenzbilder von bekannten Autokennzeichen vorliegen, so macht es u.U. keinen Sinn, jedes Zeichen zu erkennen. Wenn gewünscht, können die Segmente nach allen o.g. Filterprozessen zu einem „großen“ Segment zusammengefasst werden, das dann das gesamte Autokennzeichen repräsentiert.

Welches Verfahren, Gummiband, Freeman oder Zusammenhangskomponenten, zur Segmentierung eingesetzt werden soll, kann hier ebenfalls einstellt werden.

Die Segmentierung ist ein u.U. langwieriger Vorgang, da er direkt von der Größe des zu segmentierenden Bereichs abhängt. Es kann daher sinnvoll sein, während des Segmentierens eine Fortschrittsanzeige zu aktivieren (anstatt nur der einfachen Sanduhr), um zu sehen, wie weit die Segmentierung fortgeschritten ist.

3.9.7 Ausrichtung

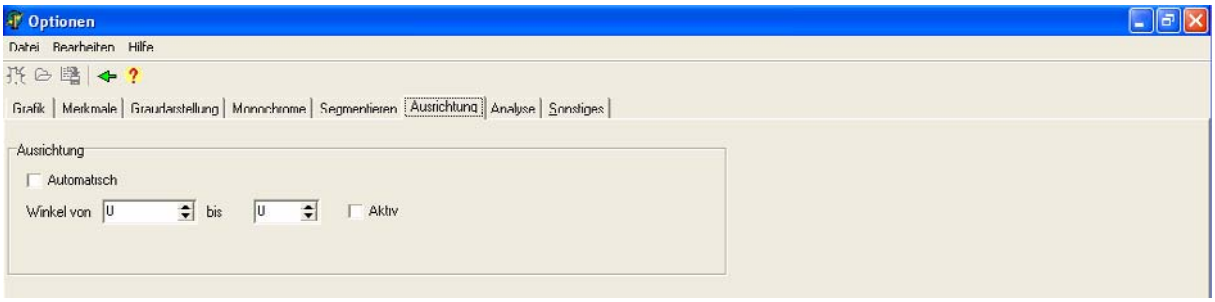


Abbildung 69 Einstellungen zur Ausrichtungsoption

Unter optimalen Bedingungen wird das Originalfoto „rechtwinklig“ aufgenommen; d.h. das Autokennzeichen wird irgendwo waagerecht im Bild vorkommen.

Sollte dies nicht der Fall sein, so besteht die Möglichkeit, das Originalfoto so zu drehen (auszurichten), dass das Autokennzeichen waagerecht wird.

Dazu wird zunächst „ganz normal“ segmentiert. Anschließend wird für die (x,y)-Koordinaten eines jeden Segments eine lineare Regression durchgeführt. Damit bekommt man die „minimale“ Steigung aller Segmente. Dreht man nun das Originalfoto um den negativen Winkel dieser Steigungsgeraden, so werden anschließend die Segmente waagerecht sein. Aus optischen Gründen lassen sich damit aber nur 2-dimensionale Drehungen der Kamera bzw. des Fahrzeugs ausgleichen.

Soll die Drehung automatisch im Erkennungsprozess durchgeführt werden, so kann dies hier aktiviert werden.

Um dabei völlig unsinnige Drehungen zu vermeiden, kann ein Min- und Maximalwert für den Drehwinkel eingestellt werden. Unter- bzw. überschreitet der berechnete Drehwinkel diese Grenzen, so wird das Bild **nicht** gedreht.

3.9.8 Analyse

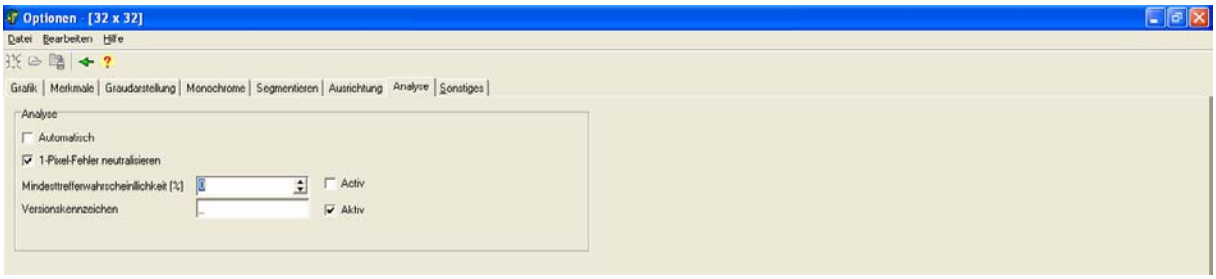


Abbildung 70 Einstellungen für die Analyseoption

Soll die Analyse automatisch nach dem Laden eines Fotos gestartet werden, so kann dies hier aktiviert werden. Auf Wunsch wird vorher eine 1-Pixel-Fehler-Korrektur durchgeführt.

Jedes gefundene Segment wird mit den gelernten Referenzen verglichen, und es wird eine relative Übereinstimmung in % ermittelt. Ist die Mindestgüte aktiv, so werden alle Übereinstimmungen kleiner dieser Mindestgüte nicht weiter untersucht.

Die gelernten Referenzen erhalten intern einen Dateinamen (z.B. A.bmp für ein gelerntes A). Werden nun je Zeichen mehrere Referenzen gelernt, so müssen diese Referenzen unterschieden werden können. Am einfachsten geschieht dies im Dateinamen der Referenz (z.B. A_1.bmp, A_2.bmp etc. wenn mehrere „A“ gelernt wurden). Beim Lernen aus Fonts wird hinter das Zeichen, getrennt durch das Versionskennzeichen (Default _) der Name der Schriftart eingetragen. Beim „Rückübersetzen“ des Dateinamens in das erkannte Zeichen, wird dann nur der Teil des Dateinamens bis zum Versionskennzeichen berücksichtigt.

3.9.9 Einstellungen

Hier können noch weitere Einstellungen durchgeführt werden:

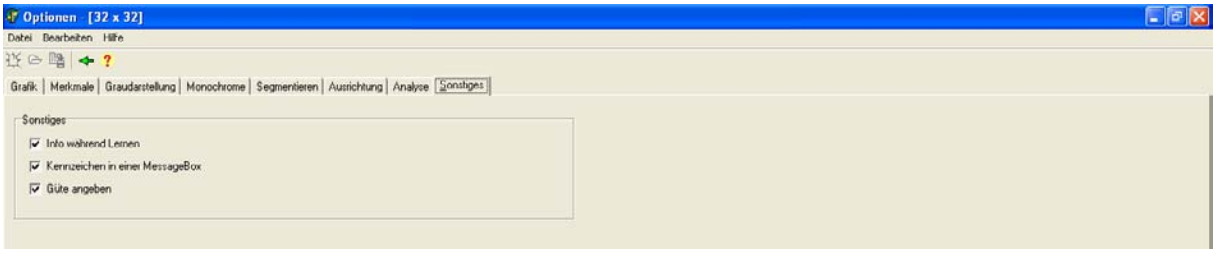


Abbildung 71 Einstellungen der sonstigen Optionen

Nach dem Start des Systems oder der Veränderung von Einstellungen, wie z.B. dem Hinzufügen/Löschen von Merkmalen oder dem Lernen/Löschen neuer Referenzen, müssen die Referenzen neu gelernt werden. Dies kann je nach Umfang der zu lernenden Referenzen länger dauern. Zur Überwachung kann eine Fortschrittsanzeige aktiviert werden.

Am Ende des Analyseprozesses wird das erkannte Kennzeichen in der Statuszeile am unteren Ende des Bildschirms angezeigt. Soll dies in einer Box zentriert auf dem Bildschirm geschehen, so kann dies hier einstellt werden.

Wird die Ausgabe der Güte als **gewogenes Mittel** der „Einzelgüten“ gewünscht, so kann dies hier eingestellt werden.

3.10 Beispiele

Hier sollen einige Beispiele Aufschluss über die Leistungsfähigkeit der SpaCAM geben.

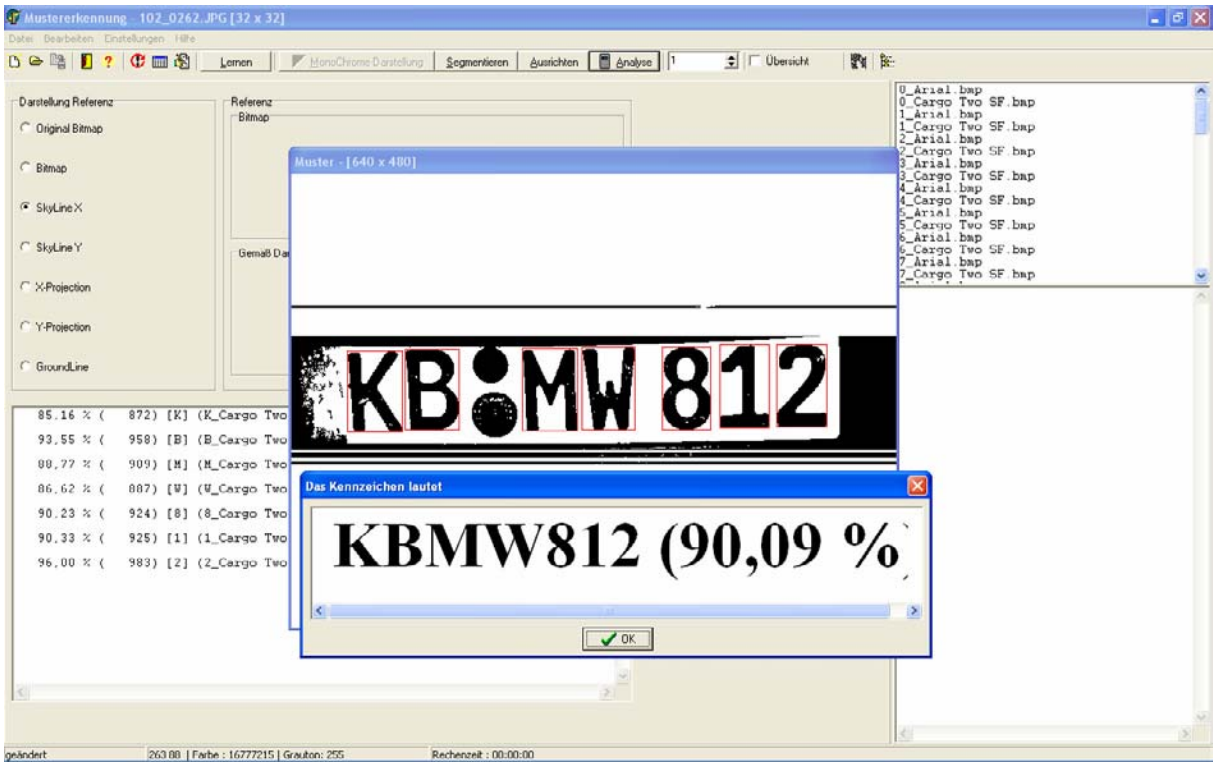


Abbildung 72 Trotz leichter Verdrehung wird das Kennzeichen erkannt

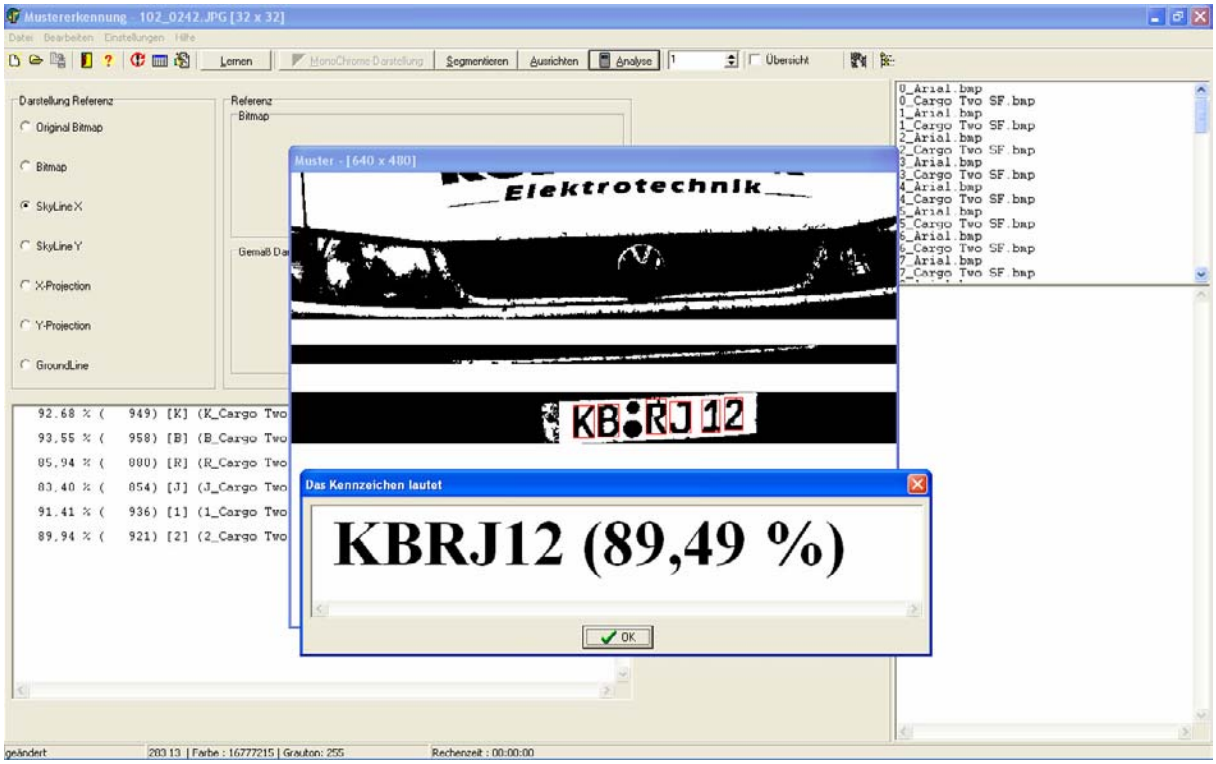


Abbildung 73 Trotz leichter Unschärfe wird das Kennzeichen erkannt

Folgendes Bild ist recht schräg aufgenommen worden.

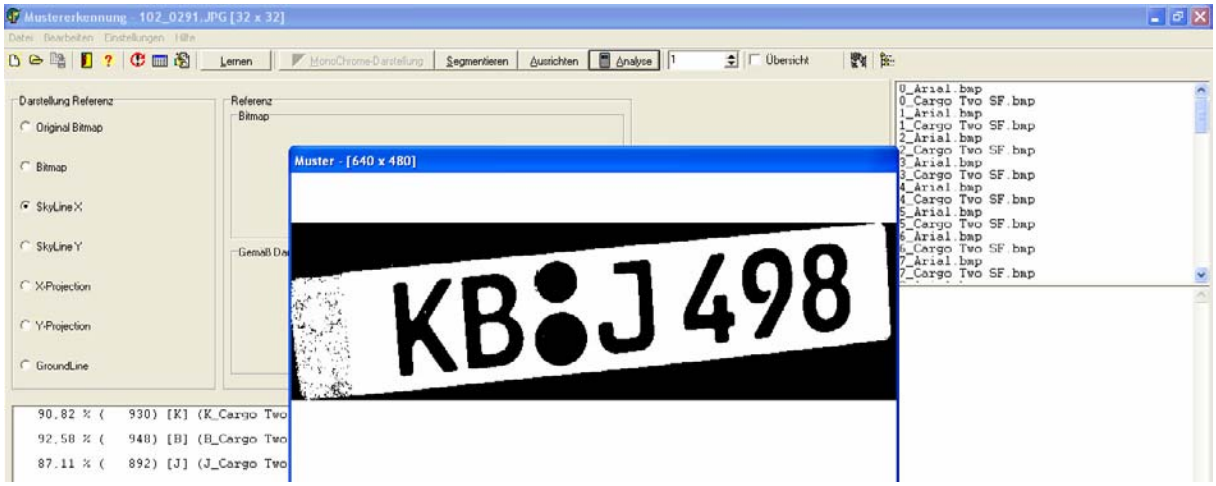


Abbildung 74 „Schräge Aufnahme“

Nach der automatischen Ausrichtung wird das Kennzeichen korrekt erkannt.

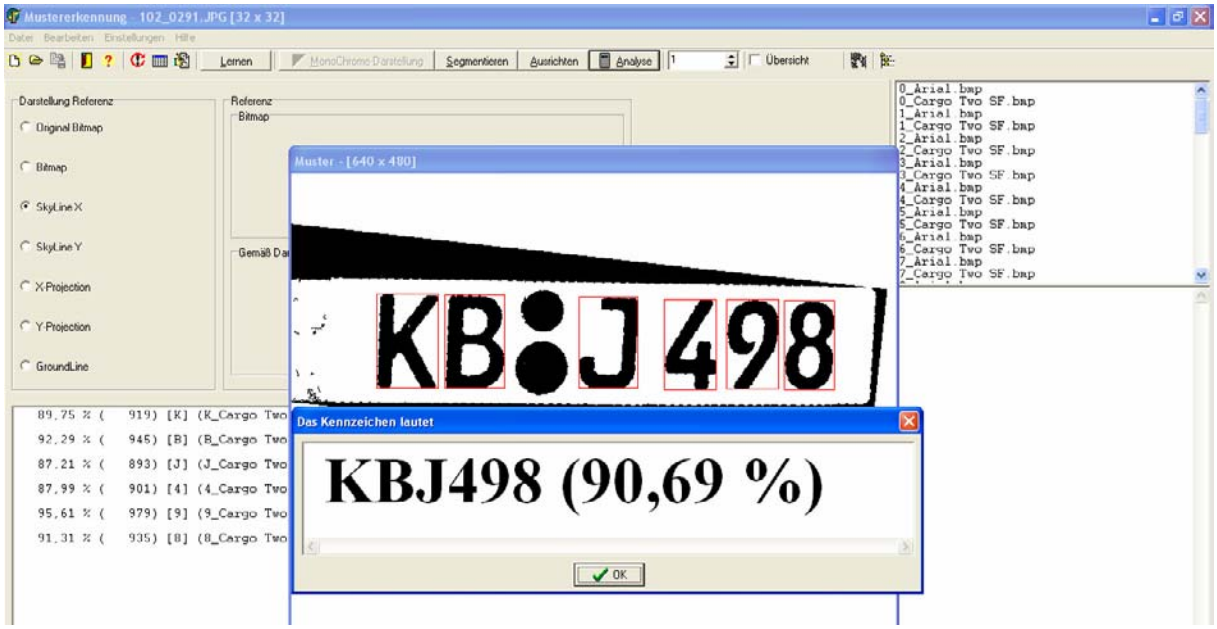


Abbildung 75 Korrekte Erkennung nach der waagerechten Ausrichtung

Auch dieses Kennzeichen mit „alter“ Schrift (Arial) wird erkannt

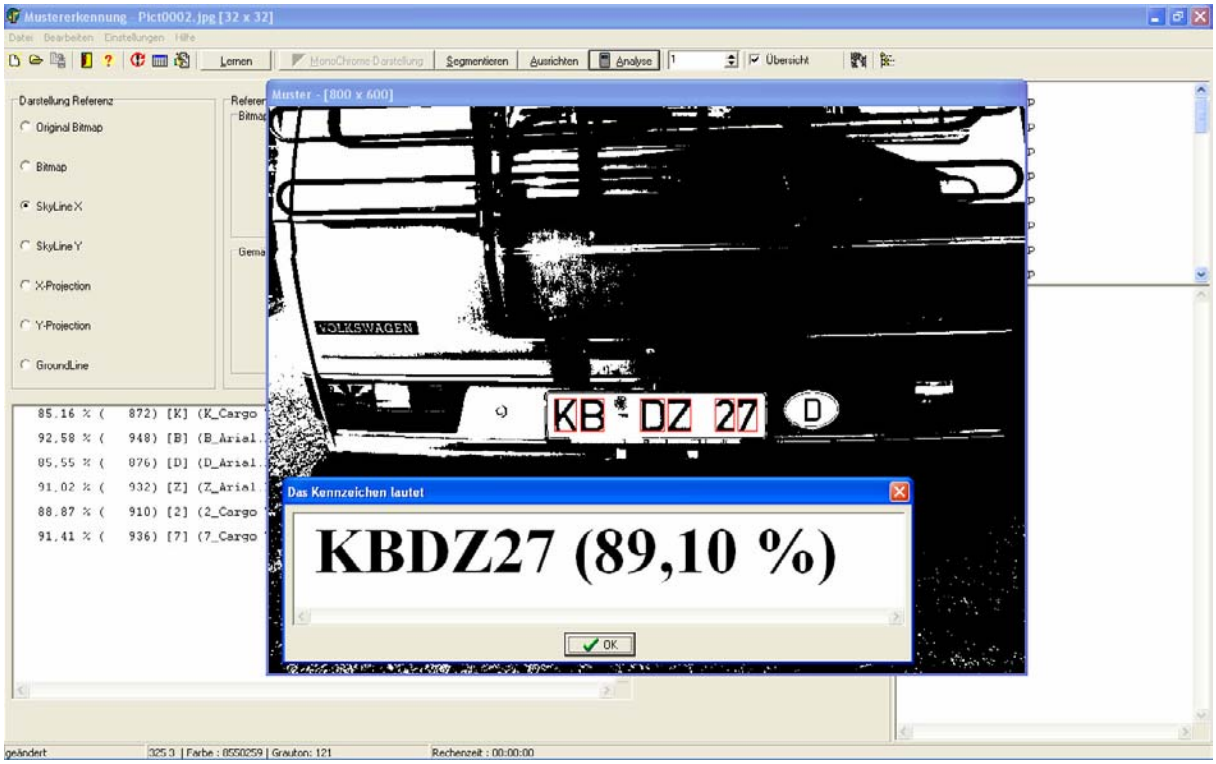


Abbildung 76 Kennzeichen mit „alter“ Schrift

Trotz einer sehr blassen Aufnahme eines mit „alter“ Schrift versehenen und durch die Waschanlage angegriffenen Kennzeichens wird fast korrekt erkannt. Lediglich die 1 wird für ein I gehalten.

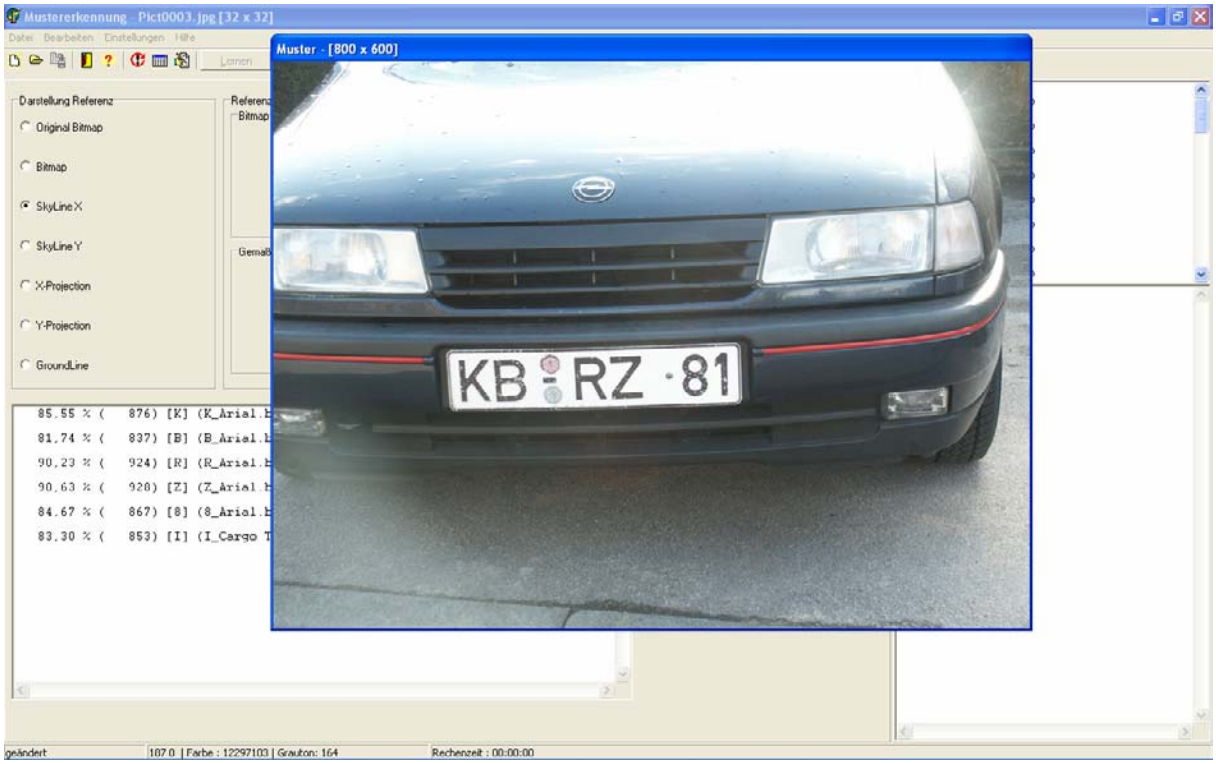


Abbildung 77 Blasse Aufnahme eines verwaschenen Kennzeichens

Abbildung 78 Erkennung bis auf die 1 korrekt

Das System erkennt, dass dieses Kennzeichen in **Deutschland** nicht möglich ist, da nach einer Ziffer kein Buchstabe mehr kommen darf.

Stellt man zwei Treffer pro Segment ein, so wird die 1 nach dem I erkannt.

Abbildung 79 Die 1 wird als zweitbester Treffer erkannt

Lässt man alle möglichen Kombinationen aus den gefundenen Treffern aufbauen, so wird das Kennzeichen richtig erkannt.

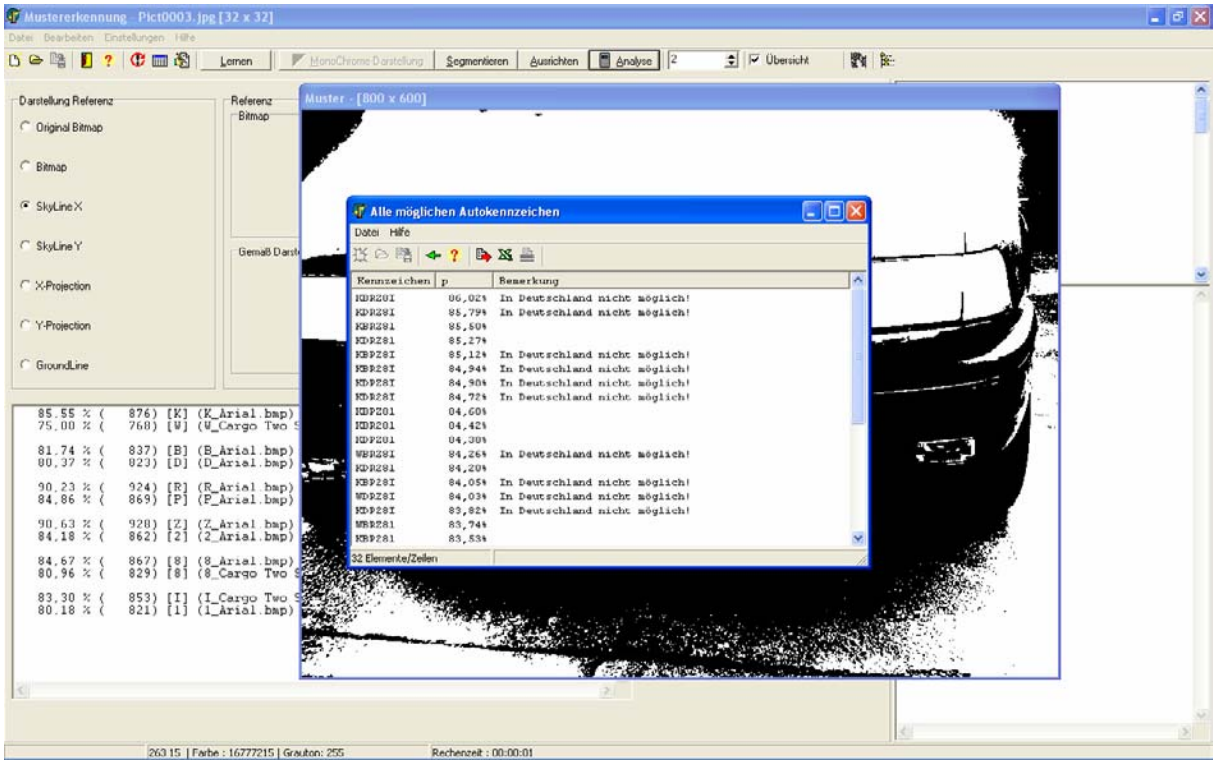


Abbildung 80 Das Kennzeichen wird richtig erkannt

3.11 Erfahrungen aus der Praxis

Die Fa. Bitzer GmbH in Hildesheim, ein Partnerunternehmen der Universität Hildesheim, war so freundlich, eine CD mit einigen Fotos aus der Praxis zur Verfügung zu stellen, um damit eine mögliche Praxistauglichkeit der als Workbench konzipierten Software zu testen. An einer kleinen Auswahl der Bilder sollen einige Aspekte der Vorverarbeitung demonstriert werden.

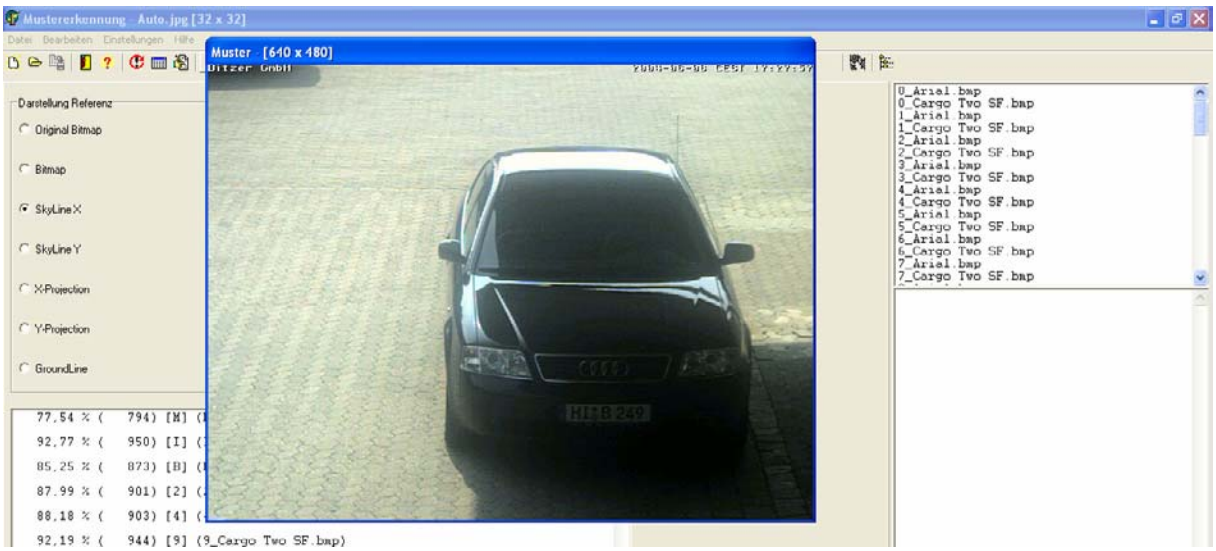


Abbildung 81 Kontrastarmes Foto

In dieser Farbaufnahme ist der Kontrast, mit dem sich das Kennzeichen abhebt, recht schwach. Selbst das menschliche Auge hat hier Probleme. Mit der Einstellung des „richtigen“ Schwellwerts kann aber das monochrome Kennzeichen deutlich sichtbar gemacht werden.

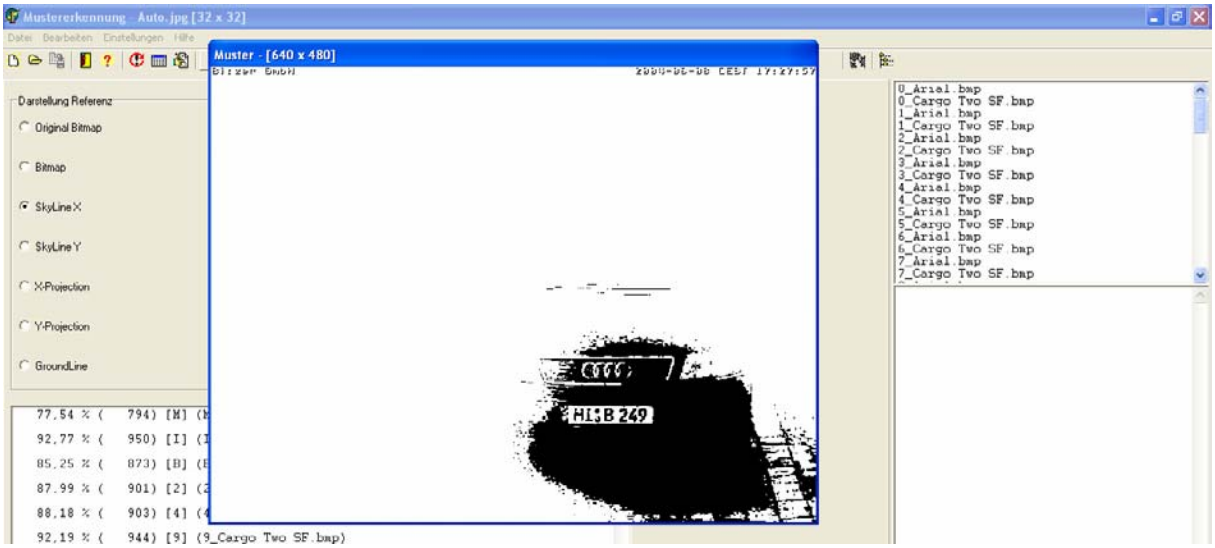


Abbildung 82 Monochrome Darstellung

Bei der Analyse kann zumindest ein großer Teil des Kennzeichens korrekt erkannt werden.

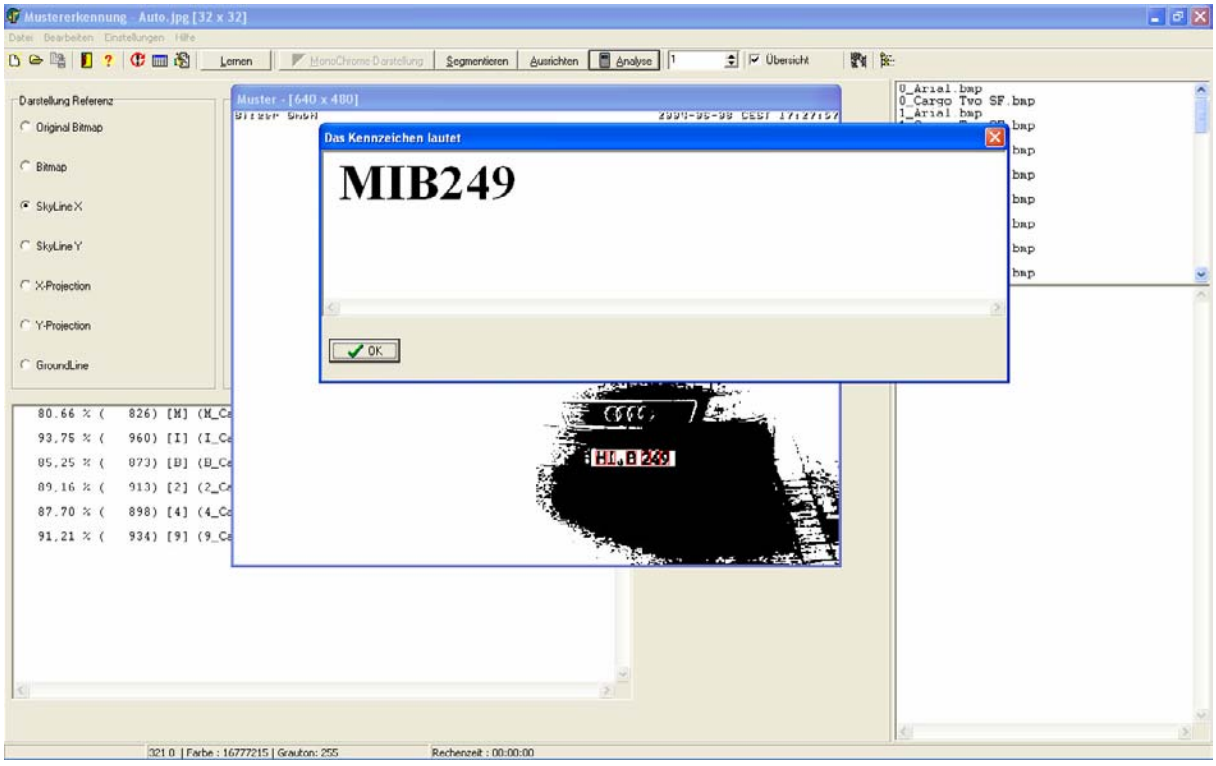


Abbildung 83 Fast korrekte Erkennung des Kennzeichens

Auf diesem Bild wird die „2“ verdeckt, so dass der in der Applikation implementierte Segmentierer sie nicht segmentieren kann.

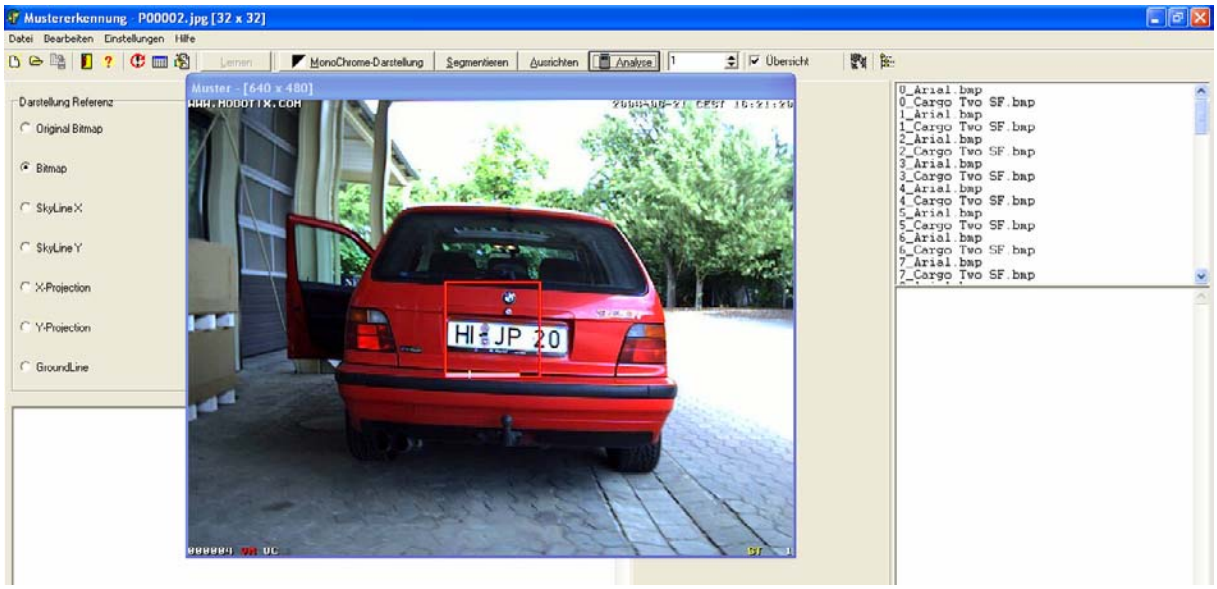


Abbildung 84 Teilweise verdecktes Kennzeichen

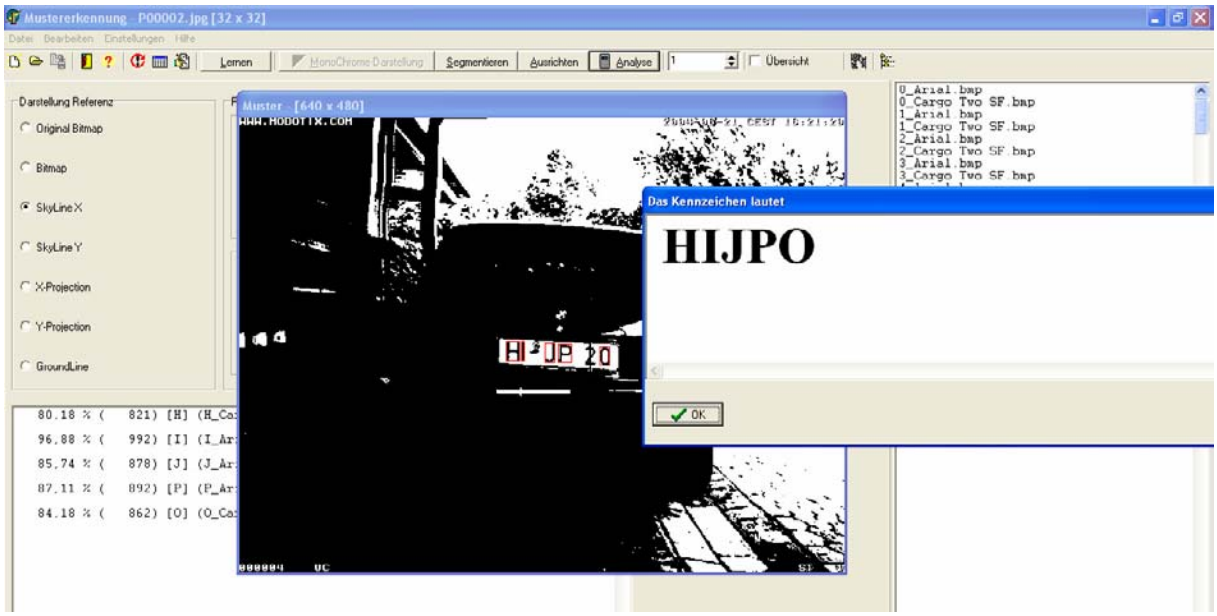


Abbildung 85 Erkennung der segmentierten Zeichen

Da die „2“ verdeckt ist, kann auch nur ein Teil des Kennzeichens erkannt werden. Dieses kann unter gewissen Umständen dennoch ausreichen, um auf das gesamte Kennzeichen zu schließen. Man hat nämlich eine Aufgabe der „Mustervervollständigung“ vor sich, wobei ein Teilmuster bereits gegeben ist. In der Praxis kann davon ausgegangen werden, dass die Sammlung der Kennzeichen aller Fahrzeuge der Firma (oder des Kundenstamms) bekannt ist, also in einer Datenbank vorliegt. Bei

solchen Problemen hat sich die SpaCAM Technologie bewährt, man könnte hier also auf bereits bekannte Codierungen zu dieser Art der Mustererkennung zurückgreifen. So etwa würde der Teilstring „HIJP0“ zum Beispiel unter Einsatz der „Paarcodierung“ siehe [**HAG1996**], zu dem gewünschten Kennzeichen führen.

Folgendes Fahrzeug hat einen recht großen Abstand zur Kamera und wurde von leicht seitwärts aufgenommen. Dadurch ist das Kennzeichen durch die Zentralprojektion nicht waagerecht. Außerdem sind Blendeffekte durch die starke Sonneneinstrahlung auf dem Kennzeichen sichtbar. Auch hier lässt sich das Kennzeichen durch die geeignete Wahl eines Schwellwertes aber wiederum gut nach monochrom wandeln. Mit Hilfe der o.g. linearen Regression lässt sich dann die leichte Drehung des Kennzeichens so korrigieren, dass es waagerecht ist. Damit ist eine Erkennung wieder möglich.

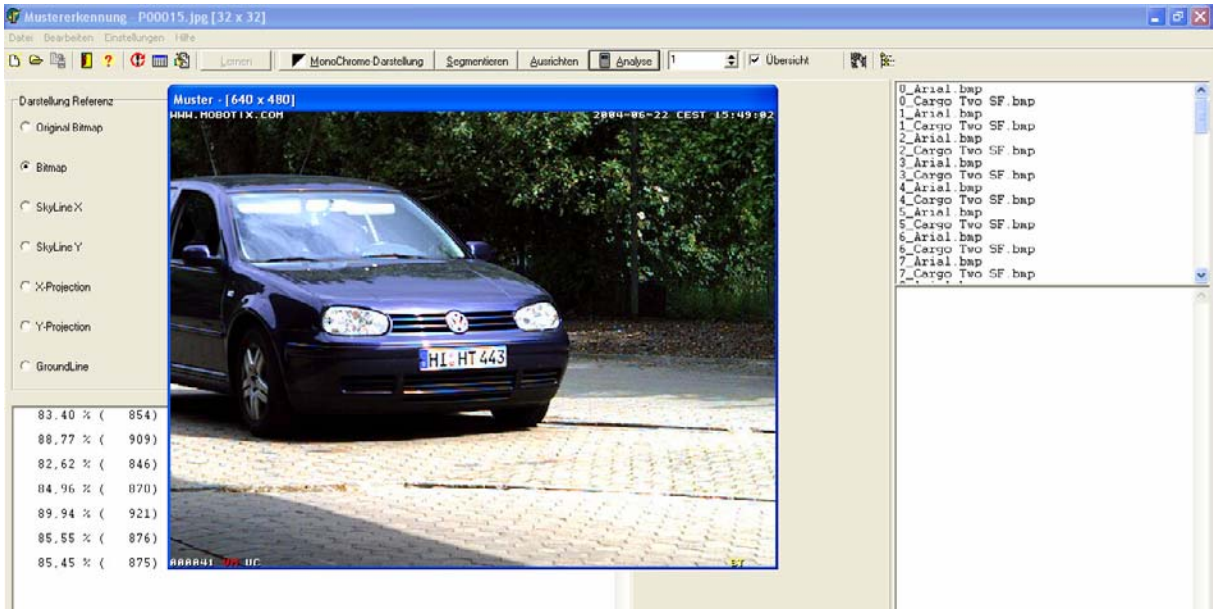


Abbildung 86 Seitliche Aufnahme

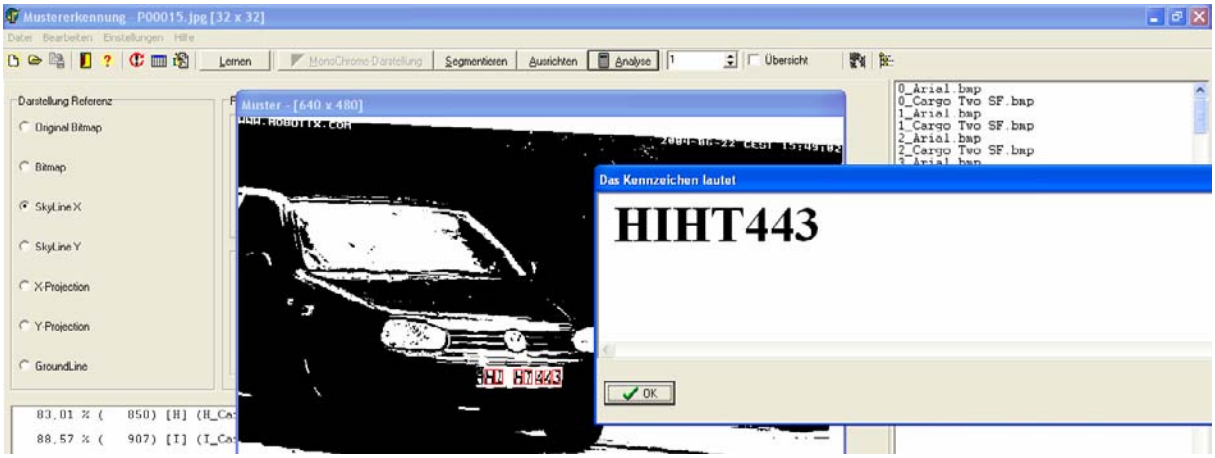


Abbildung 87 Erkennung möglich

Bereits diese kleine Auswahl an Bildern dürfte zeigen, dass die Güte der Erkennung nicht nur von der Einstellung der Workbench-Parameter abhängt, sondern auch von externen Einflussgrößen. Dazu gehören die Auflösung der Kamera und damit die vorgelegte „Größe“ der zu erkennenden Zeichen. Weiter hängt die Qualität der Bilder vom Umfeld und den Außenbedingungen bei der Aufnahme ab. Die Auflösung der Bilder kann man heute einfach und kostengünstig durch die geeignete Wahl einer Digitalkamera positiv beeinflussen; ebenso ungünstige Licht- und Umgebungsverhältnisse durch eine entsprechende Beleuchtung. Auf diese hat man auch Einfluss durch die geeignete Auswahl des Standortes der Kamera.

Einen Teil der durch die Aufnahmebedingungen herrührender Störeffekte kann man auch softwaremäßig durch Aufhellung, Kontrasterhöhung, Drehung usw. ausgleichen. Die Erkennung hängt dann von der Qualität der gelernten Referenzen und der Wahl bzw. Kombination geeigneter Merkmale ab.

Man hat also eine komplexe Situation vor sich, bei der gewisse Wahlfreiheiten bei der Einstellung der Systemparameter existieren, jedoch auch systemfremde Einflüsse durch die Umgebungsparameter hereinspielen und aufgefangen werden müssen.

3.12 Algorithmen und Datenstrukturen

3.12.1 Einleitung

In diesem Kapitel soll auf ausgewählte Algorithmen und Datenstrukturen, die in der Applikation zum Einsatz kommen, eingegangen werden. Dabei wird natürlich intensiv auf die in Delphi implementierten Datenstrukturen zurückgegriffen, es wurden aber auch einige neue Datenstrukturen und Objekte definiert.

3.12.2 Standardobjekte

Immer wieder müssen Koordinaten von Punkten P(x,y) verwaltet werden. Delphi definiert dazu den Typ ***TPoint*** als ***record***, der die beiden Variablen x und y im Integer-Format enthält.

```
type
  TPoint = record
    x, y : integer;
  end;
```

Programmcode 1 Definition von Koordinaten

Auch das Rechteck, z.B. als Bildausschnitt, wird benötigt. Delphi definiert auch hier einen eigenen varianten Typ **TRect**:

```
type
  TRect = record
    case Integer of
      0: (Left, Top, Right, Bottom: Integer);
      1: (TopLeft, BottomRight: TPoint);
  end;
```

Programmcode 2 Datenstruktur für ein Rechteck

Damit kann auf ***TRect*** sowohl über seine Eck-Koordinaten einzeln, als auch über ***TPoint*** auf die linke-obere bzw. rechte-untere Ecke zugegriffen werden.

Das Programm macht intensiv Gebrauch von Listen (vgl. zu diesem gesamten Kapitel [**DELPHI5**]). Dazu stellt Delphi das Objekt ***TList*** zur Verfügung. In dieser Liste können als Elemente alle aus ***TObject***, dem Basisobjekt von Delphi, abgeleiteten Datenstrukturen abgelegt werden. Und das nicht je Liste getrennt, sondern jedes Element einer Liste kann, dank Polymorphie, ein anderes Objekt beinhalten. Es stehen Methoden zum Hinzufügen (***add***), Löschen (***delete***), Sortieren (***sort***) usw. zur Verfü-

gung. Auf jedes Listenelement kann direkt über den Index zugegriffen werden. Damit verhält sich die Liste wie ein Array, ist aber wesentlich eleganter.

Eine „spezielle“ Liste wird benutzt, wenn lediglich Strings verwaltet werden sollen. Dazu stellt Delphi den Typ ***TStringlist*** bereit, der zusätzliche Methoden, wie zum Prüfen auf doppelte Einträge, Sortieren etc. enthält.

Wenn die Datenstrukturen weniger komplex sind (z.B. nur Ganzzahlen oder Fließkommazahlen) ist die Liste zu aufwändig. Dort bieten sich natürlich Vektoren in Form von „***Arrays of ...***“ an. Auch Matrizen sind als ***„Array of Array of ...“*** einfach abzubilden. Delphi bietet über den statischen Typ (z.B. ***A : array[0..10] of integer***), wobei in A fest 11 Integer-Werte gespeichert werden können, hinaus den dynamischen Typ (z.B. ***array of integer***) an. Dort wird der benötigte Speicherplatz zur Laufzeit mit der Standardprozedur ***SetLength*** reserviert***.*** So reserviert z.B. ***Setlength(A, 11)*** im Vektor ***A : array of integer*** genau 11 Speicherplätze für Integer-Werte, auf die wie bei den statischen Arrays über z.B. ***A[5]*** direkt zugegriffen werden kann, wobei der Index **immer** bei 0 beginnt. Solche Vektoren lassen sich jederzeit vergrößern, wobei die bisher gespeicherten Werte beibehalten bleiben, aber auch verkleinern, wobei natürlich die „abgeschnitten“ Werte verloren gehen.
Dabei sind auch mehrdimensionale Arrays möglich (z.B. definiert ***A : array of array of integer***) einen 2-dimensionale Matrix von Integer-Werten. Auf die Werte kann, wie bei statischen Typen mit z.B. ***A[5,5]*** zugegriffen werden. Haben alle Matrixzeilen die gleiche Anzahl von Elemente, so kann mit z.B. ***SetLength(A, 10, 10)*** eine 10 x 10 Matrix A generiert werden. Selbst eine unterschiedliche Anzahl von Elementen je Zeile ist machbar.

Beispiel:

```
var
  A : array of array of integer;
begin
  SetLength(A,2);           // reserviert zwei Zeilen
  SetLength(A[0],10);       // 0. Zeile hat 10 Elemente
  SetLength(A[1],20);       // 1. Zeile hat 20 Elemente
...
```

Programmcode 3 Dynamische Strukturen zur Abbildung von Matrizen

Auf diese Weise sind auch höher dimensionale Strukturen möglich. Selbstverständlich sind dabei auch Arrays von ***records*** möglich; so kann z.B. über ***var A: array of TRect*** ein Vektor von Rechtecken oder über ***var A: array of TPoint*** ein Vektor von Punkten definiert werden.

Ein spezieller und in dieser Arbeit häufig benötigter Datentyp ist der binäre Vektor. Natürlich ließe sich dies über ein ***array of boolean*** realisieren. Die interne Datenstruktur von ***boolean*** belegt aber nicht nur ein Bit, sondern standardmäßig ein Byte [**DELPHI5**], wobei true mit 11111111_2 und false mit 0 beschrieben wird. Somit handelt es sich bei einem ***array of boolean*** intern um ein ***array of byte*** mit entsprechender Verschwendung von Speicherplatz.

Aus diesem Grund liefert Delphi das Objekt ***TBits*** mit, in dem intern jeweils 32 Bits in Integer-Werten gespeichert werden. Über die Kapselung des Objekts und entsprechende Methoden kann man aber bei einer Variablen ***V:TBits*** über ***V[i]*** auf die i. Komponente des Binärvektors V lesend und schreibend zugreifen. ***V.Size*** liefert die Größe des Vektors V.

Wie bereits erwähnt, ist das Zentralelement für die graphische Darstellung der Muster, Bilder etc. das Bitmap. Delphi stellt dazu den Typ ***TBitmap*** zur Verfügung, der eine Fülle von Eigenschaften und Methoden besitzt. Eine der wichtigsten ist die Zeichenfläche (canvas), über die die ***property Pixels[x, y:integer]:TColor*** auf jeden Bildpunkt über die x und y Koordinate lesend und schreibend zugreifen kann. Beim Lesen ist der Rückgabewert die Farbe im **RGB**-Format als Typ TColor des entspre-

chenden Bildpunkts. Zum Setzen kann der Farbwert ebenfalls im **RGB**-Format als ***TColor*** angegeben werden.

Beispiel:

```
Var
  B : TBitmap     // Bitmap
  C : TColor      // Farbwert
begin
…
  B.canvas.pixels[5,4]:=clWhite;
.....
  C:=B.canvas.pixels[10,10]
…
```

Programmcode 4 Zugriff auf die Bildpunkte eines Bitmaps

Mit verschiedenen Methoden kann auf die Zeichenfläche gezeichnet werden. Dazu werden die graphischen Grundfunktionen zum Zeichnen von Linien (***lineto***), Ellipsen/Kreise (***ellipse***), Rechtecken (***FrameRect***) etc. bereitgestellt.

Nicht nur Bitmaps haben Zeichenflächen, sondern auch viele andere Objekte. So gibt es z.B. die sichtbare ***TPaintBox***, auf die nicht sichtbare Objekte wie das Bitmap oder das allgemeinere Graphikobjekt ***TGraphic*** (aus dem TBitmap vererbt wurde) gezeichnet werden können. Dazu gibt es zwei zentrale Methoden:

1. ***procedure draw(x,y:integer; Graphic:TGraphic)*** zeichnet die Graphik an die Stelle x,y der Ziel-Zeichenfläche. Sofern die Ziel-Zeichenfläche kleiner ist als Graphic, so werden die nicht sichtbaren Teile einfach abgeschnitten.
2. ***procedure StretchDraw(r:TRect, Graphic:TGraphic)*** zeichnet die Graphik in das von r umgebene Rechteck und passt die Graphik ggf. an, sollte sie größer sein.

4 Ausblick

4.1 Weitere Anwendungsgebiete

Die in dieser Arbeit benutzte Methode, Fonts zum Lernen von Referenzen zu nutzen, kann auf alle Muster erweitert werden, die über Fonts darstellbar sind. Somit ist auch die Erkennung von Texten (incl. OCR) mit dieser Software grundsätzlich möglich. Dies schließt natürlich auch viele weitere Kennzeichen ein, bei denen die Schrift über irgendwie definierte Fonts entstehen; z.B. Container, Lokomotiven, Waggons, Flugzeuge etc.

Kfz-Kennzeichen sind monochrome Objekte. Interessant wird sicherlich auch sein, mit grauen oder farbigen Objekten zu arbeiten. Dann könnten z.B. die Logos auf LKW erkannt werden, um ggf. daraus den Eigentümer der Spedition zu identifizieren.

Bislang wurden nur 2-dimensionale Muster angesprochen. Die Welt ist aber 3-dimensional. Somit liegt es nahe, auch im Raum Muster zu erkennen. Der wohl einfachste Fall liegt vor, wenn das Objekt nicht rechtwinklig zur Kamera liegt, also z.B. das Fahrzeug schräg von der Seite oder von oben aufgenommen wird. Bei leichten Verdrehungen ist die Erkennungsrate bereits mit diesem Programm recht hoch. Sind beim normalen Fotografieren aufgrund der Zentralprojektion starke Verzerrungen zu erwarten, könnte man mit Hilfe von 3D Kameras und der Weiterentwicklung der hier beschriebenen Techniken versuchen, Mustererkennung durchzuführen.

Aber auch die Lageerkennung von in einer Kiste liegenden Bauteilen gleicher, aber auch ungleicher Bauart durch einen Roboter dürfte ein lohnendes Forschungsgebiet sein.

4.2 Ideale virtuelle Zeichen

Im Kontext der Merkmalsgenerierung und –analyse ließe sich die Workbench noch leicht so modifizieren, dass auch eine gewisse Auswahl oder Sammlung von Merkmalen (Merkmalskombinationen) zur Klassifikation und Erkennung definiert werden könnte, ohne ein natürliches (externes) Zeichen haben zu müssen (dies geht über die oben beschriebene Gruppierung noch hinaus). Es würden dann aus der Bildeingabe kommende Merkmalsvektoren mit diesen „idealen“ Kombinationen verglichen, um eine Entscheidung zu finden. Das kam hier bei der Aufgabenstellung zu Kfz-Kennzeichen zwar nicht vor, da auf einen gegebenen Referenz-Font zurückgegriffen werden konnte, kann aber in einem anderen Zusammenhang sinnvoll sein. Der SpaCAM-Technik ist dies egal, sie benötigt ja „nur“ die Merkmalsvektoren. Insofern liegen hier weitere Forschungsfragen vor, wobei sich der Schwerpunkt der Untersuchung auf Erkennungsprobleme verlagert, bei denen Objekte keine vorgegebene Referenz haben. Beispiele sind Unterschriften, akustische Signale, Bilder der Medizin, etc.

4.3 Zusammenfassung und Fazit

4.3.1 Merkmale für die SpaCAM

Die Darlegungen der Arbeit zeigen, dass die Technologie der spärlich codierten Assoziativmatrizen geeignet ist, das Problem der Erkennung von Autokennzeichen (stehender Fahrzeuge) erfolgreich anzugehen. Man muss dazu auf jeden Fall passende Merkmale finden und sie in das Format der SpaCAM bringen. Wie dieses für spezielle Bildmuster, z.B. Autokennzeichen, geschehen kann, welche Aufwände bzw. Probleme man damit hat und wie man diese beheben kann, wurde an verschiedenen Beispielen in dieser Arbeit erstmals gezeigt. Eine entscheidende Aufgabe bestand darin, geeignete Merkmale zur Beschreibung der in Frage kommenden Eingangsdaten zu finden und zu entwickeln. Diese Merkmalsmenge ist von vornherein weder festgelegt noch eindeutig. Man könnte hier bei der aktuellen Mustererkennungsaufgabe sicher noch weitere finden, die vielleicht ebenso geeignet sind wie die besprochenen und analysierten. Für den echten Einsatzfall muss man die Vorgestellten vermutlich noch überarbeiten oder ergänzen. Es wurde hier aber bewusst ein Um-

fang an Merkmalen entwickelt, der zumindest im Grundsatz die Vielfalt der Phänomene aufzeigen ließ. Zum Beispiel die „SkyLine Y“, die gut diskriminierte, im Vergleich zur „Stabilität“, die zur Interpretation der hier zu erwartenden Eingangsdaten praktisch ungeeignet war. Bei anderen Erkennungsaufgaben, z.B. bei Firmenlogos, werden einige der hier analysierten Merkmale an Bedeutung verlieren, andere etwas dazu gewinnen. Für Modifikationen dieser Basismenge stehen Softwaretools in der Workbench zur Verfügung.

4.3.2 Experimentierumgebung

Die für das Ziel der Merkmalsanalyse entwickelte Workbench enthält erstmals alle Funktionalitäten, die nötig sind, um solche Untersuchungen durchzuführen und die verwertbaren Parametereinstellungen zu finden. Sie enthält noch weitere, ganz andersartige Funktionen zur Bearbeitung des Datenmaterials. Das ist zwar nur ein „Detail“, aber für den Ablaufprozess unverzichtbar. Da das Bildmaterial aus der Realität die unterschiedlichsten Abweichungen bzw. Fehler im Vergleich zur idealen Vorlage haben kann, muss man Hilfsmittel entwickeln, die die konkreten Gegebenheiten sichtbar machen und messen lassen. Obwohl man hier in vielen Bildbearbeitungsprogrammen Vorlagen findet, wurden die wichtigsten Funktionen selbst entwickelt. Erstens kommt man nicht an die fremden Einzelmodule heran, teilweise wäre das auch kostenpflichtig, zweitens sollten alle Funktionen der Workbench aufeinander abgestimmt und „homogen“ sein. Das erleichtert die Pflege des Gesamten und die Modifikation von Teilen.

4.3.3 Datenbankabgleich

Die SpaCAM-Technologie wurde hier erstmals mehrfach in einen (Bild-) Erkennungskontext eingebunden, wobei die Erkennungsaufgaben gravierend unterschiedlicher Art waren, mithin die Merkmalsmengen und die Codierung ganz verschieden ausfielen. Im ersten Teilprozess ging es um die Zeichenerkennung (Bild nach ASCII), im zweiten Teilprozess um das Zuweisen der erkannten Buchstaben-Ziffernfolge zu dem Kraftfahrzeug in der Datenbank. Es kann ja vorkommen, dass gewisse, nicht vorhersagbare Fehler oder Ungereimtheiten in den von der Bilderkennung gelieferten Kennzeichen-Vorschlägen sind, die einen konventionellen Abgleich mit den Datenbankeinträgen erfolglos lassen. Die eigentliche Retrieval-Aufgabe bei diesem zweiten

Schritt war jedoch bereits durch frühere Arbeiten, z.B. [Hag1996], gut untersucht, so dass hier auf schon bewährte Lösungen zurückgegriffen werden konnte. Durch diese Kombination von Funktionseinheiten lässt sich die Erkennungsleistung insgesamt leichter und transparenter verbessern als durch überhöhten Aufwand in den Einzelprozessen. Solche Kombinationen finden sich bislang weder in der Literatur, noch in den marktgängigen Anwendungen.

4.3.4 Praxistauglichkeit

Obwohl die Entwürfe und Untersuchungen von der Aufgabenstellung her den Verarbeitungsprozess über die SpaCAM-Technik als Fokus hatte und nicht den Wettstreit mit marktgängigen Systemen, war es natürlich von Interesse zu prüfen, wie weit man mit den entwickelten Einsichten und prognostizierten Parametern in einer praktischen Situation kommt. Die Erfahrungen mit vielen Fotos unterschiedlichster Herkunft und Qualität haben – auch dank der Bilder der Fa. Bitzer GmbH – gezeigt, dass tatsächlich die Erkennungsrate den (hohen) Erwartungen entspricht, durchgängig stabil erscheint und bereits in der vorliegenden Ausbaustufe sogar kommerziellen Ansprüchen gerecht werden könnte. Die Durchsicht von einigen im Markt angebotenen Produkten (www.vmt-duessel.de in Essen mit dem TrafficScan-System [**TRAFFICSCAN**], www.axxteq.de in Aachen mit einem LPR-Kennzeichenleser [**AXXTEQ**], www.eltec.de in Mainz mit dem ELTEC EL-RECO System [**ELTEC**_a] bzw. [**ELTEC**_b], www.vitronic.de in Wiesbaden mit dem PoliScan-System [**POLYSCAN**], nur um einige Beispiele zu nennen) ergab keine Hinweise auf neuartige Techniken, wie sie in dieser Arbeit untersucht wurden. Leider lassen diese Firmen aus Wettbewerbsgründen keine Einsicht in die Interna der verwendeten technologischen Spezifika zu. Jedoch werden dort meistens Infrarotkameras eingesetzt, deren Eigenarten für die Ausgangsproblemstellung hier allerdings nicht untersucht worden sind. Eine sinnvolle Folgeaufgabe aus dieser Arbeit könnte daher die pilotweise Erprobung der gewonnenen Erkenntnisse in einer konkreten Anwendersituation sein.

Literaturverzeichnis

[AXXTEQ] Access Technology for the new Millennium; MOBOFIX Praxisbericht; Fa. Axxteq GmbH; D-52146 Würselen; http://www.axxteq.de/ und www.mobotix.de/mx_pdf/mobotix_praxis_Fendt_D.pdf

[BECK1997] Tim Becker, „Mustererkennung mit SpaCAM am Beispiel von Signaturen“, Diplomarbeit, Universität Hildesheim, Institut für Mathematik, 1997

[CC2000] CC2000-Lösungen mit automatischer KFZ-Kennzeichenerkennung, Fa. CC2000 Technologies GmbH; D-49457 Drebber; www.cc2000.net/CC2000-Prospekte/CC2000-KFZ-Kennzeichenerkennung.pdf

[DELPHI5] Handbücher, Hilfedateien etc zu Delphi 5.0 Professional

[DIET2003] Kai Diethelm, „Bildverarbeitung“, Kapitel 3.2, TU Clausthal 2003/04, www-public.tu-bs.de:8080/~diethelm/lehre/bv03/bv03.html

[DIET2003a] Kai Diethelm, „Bildverarbeitung“, Kapitel 4, TU Clausthal 2003/04, www-public.tu-bs.de:8080/~diethelm/lehre/bv03/bv03.html

[ELTEC_a] EL-RECO; ELTEC´s license plate recognition system; Fa. ELTEC Elektronik AG; D-55129 Mainz; http://www.eltec.de/

[ELTEC_b] Das Wesentliche im Blick; Verkehrstechnik; Fa. ELTEC Elektronik AG; D-55129 Mainz; http://www.eltec.de/

[FREY] Jochen Frey, „Bildsegmentierung mit der Topologischen Merkmalskarte in MR- und CT-Bildern“, mbi.dkfz-heidelberg.de/mbi/TR/TR49/dipl.bch.html

[GONZ2002] Rafael C. Gonzalez, Richard E. Woods, „Digital Image Processing“, Prentice Hall International, 2002

[HAG1996] Michael Hagström, „Textrecherche in großen Datenmengen auf der Basis spärlich codierter Assoziativmatrizen“, Dissertation, Universität Hildesheim, Fachbereich Mathematik, Informatik und Naturwissenschaften, 1996

[HEIT1994] Michael Heitland, „Einsatz der SpaCAM-Technik für ausgewählte Grundaufgaben der Informatik“, Dissertation, Fachbereich für Mathematik, Informatik und Naturwissenschaften der Universität Hildesheim, 1994

[HES2003] Pressemitteilung des hessischen Innenministers Volker Bouffier zur Novellierung des Hessischen Polizeigesetzes (HSOG) vom 19.11.2003, http://www.hessen.de

[JAEHN2002] Bernd Jähne, “Digital Image Processing, w. CD-ROM“, Berlin, Springer-Verlag, 2002

[JAEHN2002a] Bernd Jähne, „Digitale Bildverarbeitung, m. CD-ROM“, Berlin, Springer-Verlag,

2002

[JENSCH] P. Jensch, „Mustererkennung (Script/Loseblattsammlung) BS Bildsegmentierung“,condor.informatik.uni-oldenburg.de/lehre/WS01-SigBildVer/6_Bildsegm.pdf

[KENNZ] Informationen über alte und neue Autokennzeichen, www.kennzeichen.org

[LUBA2001] Thomas Luba, „Konkordanzen in langen Zeichenketten“, Dissertation, Hildesheimer Informatikberichte, Institut für Mathematik und Angewandte Informatik, 2001

[LUEN] Alexander Lüning, Kantenerkennung und Skelettierung, page.mi.fu-berlin.de/~luening/CoVi/Kantendetektion.html

[MART1997] Christian Marten, „Entwurf einer ICR-Arbeitsumgebung mit Testumgebung für Assoziativspeicher“, Diplomarbeit, Universität Hildesheim, Institut für Mathematik, 1997

[NEU1993] Klaus Neumann, Martin Morlock, „Operations Research“, München/Wien, Carl Hanser Verlag, 1993

[PETRY1999] Nikolaus Petry ,”Fuzzy Logik und neuronale Netze“, Internet-Zeitschrift für Rechtsinformatik, JurPC Web-Dok. 187/1999, Abs. 1 – 54, www.jurpc.de/aufsatz/19990187.htm

[PLA1986] Plate, J., „Computergraphik: Einführung – Algorithmen – Programmentwicklung“, München, Franzis-Verlag, 1986

[POLYSCAN] **PolyScan**$^{\textbf{speed}}$ – Neue Wege der digitalen Geschwindigkeitsüberwachung; Fa. Vitronic Bildverarbeitungssysteme GmbH; D-65189 Wiesbaden; http://www.vitronic.de

[PRATT2001] William K Pratt, „Digital Image Processing, w. CD-ROM“, Wiley & Sons, 2001

[RITT1991] Helge Ritter, „Neuronale Netze: eine Einführung in die Neuroinformatik selbstorganisierter Netzwerke“, Bonn, München, Addison-Wesley, 1991

[SCHÜ1977] Jürgen Schürmann, „Polynomklassifikatoren für die Zeichenerkennung: Ansatz, Adaption, Anwendungen - 1. Auflage - , München, Wien: Oldenbourg, 1977

[SUND1999] Thomas Sundermann, “Computer Vision” Bildverbesserung durch Glättung der Grauwerte – Digitale Filteroperationen im Ortsbereich“, FU-Berlin, Informatik (SS1999), page.mi.fu-berlin.de/~sunder/ausarbeitung1/ausarbeitung1.html

[TRAFFICSCAN] TrafficScan: Identifizierung von beweglichen Fahrzeugen; Fa. VMT-Düssel; D-45356 Essen; http://www.vmt-duessel.de/

[WEV1998] Klaus Wevelsiep, „Verfahren zur Segmentierung von Schriftzeichen und Symbolen auf Kfz-Kennzeichenschildern und formatierten Datenträgern sowie zur Segmentierung von Mustern in komplexen Szenen“, Deutsches Patentamt, Offenlegungsschrift DE 197 17 814 A 1, 1998

[WiWo2003a] Wirtschaftswoche, Nr. 31/2003; 27.7.2003

[WiWo2003b] Wirtschaftswoche, Nr. 35/2003; 21.8.2003

[WiWo2003c] Wirtschaftswoche, Nr. 51/2003; 11.12.2003

[WiWo2004a] Wirtschaftswoche, Nr. 14/2004; 25.3.2004

[WiWo2004b] Wirtschaftswoche, Nr. 20/2004; 6.5.2004

[YAP] Yap Keem Siah, Tay Yong Haur, Marzuki, Khalid, Tahir Ahmad; Centre for Artificial Intelligence and Robotics (CAIRO), Faculty of Electrical Engineering, University Technology Malaysia, Jalan Semarak, 54100 Kuala Lumpur; “Vehicle License Plate Recognition by Fuzzy Artmap Neural Network”, www.cairo.utm.my/publications/ksyap_wec99.pdf

Anhang

Die für die Arbeitsumgebung notwendigen Erläuterungen zu den Objekten, Datenstrukturen und zum Programmablauf sind separat zusammengestellt und befinden sich in elektronischer Form auf einer CD. Zusätzlich sind dort die im Literaturverzeichnis angegebenen Internetseiten abgespeichert, da nicht sichergestellt werden kann, dass zu jedem späteren Zeitpunkt die Seiten im Web noch verfügbar sind.